UFMT - Universidade Federal de Mato Grosso

Disciplina: Algoritmos e Programação II
Docente: Prof. Dr. Renan Vinicius Aranha
Data: 31/07/2024

GABARITO DO SIMULADO

---

1. Qual a diferença de passagem por valor e passagem por referência? É correto afirmar que a linguagem de programação C não possui nenhuma estratégia que possibilite superar as limitações da passagem por valor? Se sim, por quê? Se não, por quê?

   Gabarito: Diferença: Na passagem por valor, envia-se uma cópia do dado; alterações na função não afetam a variável original. Na passagem por referência, envia-se o endereço da variável, permitindo que a função altere o dado original.
   Sobre a linguagem C: É correto afirmar que C suporta nativamente apenas passagem por valor. No entanto, é **falso** dizer que não há estratégia para superar isso.
   Justificativa: Podemos simular a passagem por referência utilizando **ponteiros**. Ao passar o endereço de uma variável (valor do endereço) para uma função, podemos desreferenciar esse ponteiro dentro da função para modificar o valor original.

2. Quais diferenças existem entre a alocação estática e dinâmica de um vetor? Existe uma abordagem mais vantajosa que a outra?

   Gabarito: Alocação Estática: Tamanho definido na compilação, memória alocada na Stack. Mais rápida, mas com tamanho fixo e limitado.
   Alocação Dinâmica: Tamanho definido na execução (ex: `malloc`), memória alocada na Heap. Mais flexível, permite redimensionamento, mas exige gerenciamento manual (`free`).
   Vantagem: Não existe uma absolutamente melhor; depende do uso. Para dados de tamanho fixo conhecido, a estática é melhor (simples e rápida). Para dados cujo volume varia ou é desconhecido (como cadastros), a dinâmica é mais vantajosa.

3. Bartolomeu é um estudante de programação que está aprendendo a lidar com aritmética de ponteiros. Atualmente, o seu código está no seguinte estágio de implementação:

```c
#include <stdio.h>

int main(void) {
    int vetor[3] = {7, 900, 125};
    printf("O elemento da terceira posicao e %d\n", ________ );
    return 0;
}

```

   Qual fragmento de código deve ser inserido no espaço para que o elemento da terceira posição do vetor seja impresso corretamente?

Gabarito: Fragmento: `*(vetor + 2)`
Explicação: `vetor` aponta para o início. Somar 2 desloca o ponteiro duas posições (para o índice 2, que é o terceiro elemento). O operador `*` acessa o valor contido naquele endereço (125).

4. Agora que você ajudou Bartolomeu, ele quer ir além. Criou um ponteiro genérico que aponta para esse vetor. Mostre como ele pode usar o ponteiro genérico para imprimir o elemento que está na segunda posição do vetor.

```c
#include <stdio.h>

int main(void){
    int vetor[3] = {7, 900, 125};
    void *generico = vetor;
    printf("O elemento da segunda posicao e %d\n", ________ );
    return 0;
}

```

Qual fragmento de código deve ser inserido no espaço para que o elemento da segunda posição do vetor seja impressa corretamente a partir do ponteiro genérico?

Gabarito: Fragmento: `*((int*)generico + 1)`
Explicação: Como `generico` é `void*`, precisamos fazer o casting para `(int*)` para que o compilador saiba o tamanho do dado ao realizar a aritmética. Somamos 1 para ir à segunda posição e usamos `*` para obter o valor.

5. Analise o código a seguir, envolvendo a declaração de uma estrutura:

```c
#include <stdio.h>

struct {
    char nome[100];
    int idade;
} Pessoa;

int main(void){
    Pessoa cebolinha = {"Cebolinha", 5};
    return 0;
}

```

Apresente uma revisão crítica do fragmento de código, abordando sua eficácia e operacionalidade. Caso identifique qualquer imprecisão, ofereça uma justificativa para tal.

Gabarito: O código contém um erro conceitual e de sintaxe que impedirá a compilação correta na função `main`.
Problema: A declaração `struct { ... } Pessoa;` define `Pessoa` como uma **variável global** dessa estrutura, e não como um **tipo**.
Consequência: Na linha `Pessoa cebolinha...`, o compilador tentará usar uma variável como se fosse um tipo, o que é inválido.
Correção: Deve-se usar `typedef struct { ... } Pessoa;` para definir `Pessoa` como um novo tipo de dado.

6. Bartolomeu e seu amigo Franciscleidson estão aprendendo, juntos, a manipular arquivos em C. Bartolomeu implementou um algoritmo que escreveu um arquivo codificado e o enviou para Franciscleidson. O desafio deste era implementar um algoritmo que não apenas lesse o arquivo, mas também fosse capaz de decodificar a mensagem. No entanto, para a surpresa de Franciscleidson, o conteúdo do arquivo sumiu quando ele executou o seu programa. O que pode ter acontecido nessa situação?

Gabarito: Causa provável: Franciscleidson abriu o arquivo utilizando o modo de abertura **"w"** (write) ou **"wb"**.
Justificativa: Ao abrir um arquivo existente com a flag "w", a linguagem C trunca o arquivo (apaga todo o conteúdo) preparando-o para receber novos dados do zero. Para ler sem apagar, ele deveria ter utilizado o modo **"r"** (read).

7. Filomena está implementando um programa para controlar os livros que possui em seu acervo pessoal. Para cada livro, ela deseja armazenar as seguintes informações: título, ISBN, autor, ano de aquisição e status (lido ou não lido). Considerando este cenário, apresente:
   (a) A implementação de uma estrutura capaz de armazenar os dados desejados por Filomena;

Gabarito:

```c
typedef struct {
    char titulo[100];
    char isbn[20];
    char autor[100];
    int ano_aquisicao;
    int status; // 0: Nao lido, 1: Lido
} Livro;
```

   (b) A implementação de uma função capaz de imprimir os dados de um único livro;

Gabarito:

```c
void imprimirLivro(Livro l) {
    printf("Titulo: %s\n", l.titulo);
    printf("ISBN: %s\n", l.isbn);
    printf("Autor: %s\n", l.autor);
    printf("Ano: %d\n", l.ano_aquisicao);
    printf("Status: %s\n", l.status ? "Lido" : "Nao lido");
}
```

   (c) A implementação de uma função capaz de alterar, a partir da leitura do teclado, os dados de um único livro;

Gabarito:

```c
void alterarLivro(Livro *l) {
    if(!l) return;
    printf("Novo titulo: ");
    scanf(" %[^\n]", l->titulo);
    printf("Novo ISBN: ");
    scanf(" %[^\n]", l->isbn);
    printf("Novo Autor: ");
    scanf(" %[^\n]", l->autor);
    printf("Ano: ");
```

```
    scanf("%d", &l->ano_aquisicao);
    printf("Status (1-Lido, 0-Nao): ");
    scanf("%d", &l->status);
}
```

(d) Num cenário em que muitos livros estão cadastrados no sistema, uma função que exclui um livro do acervo pessoal de Filomena.

Gabarito:

```
// Retorna 1 se sucesso, 0 se nao encontrou
int excluirLivro(Livro acervo[], int *qtd, char *isbn) {
    for(int i = 0; i < *qtd; i++) {
        if(strcmp(acervo[i].isbn, isbn) == 0) {
            // Desloca elementos para cobrir o buraco
            for(int j = i; j < *qtd - 1; j++) {
                acervo[j] = acervo[j+1];
            }
            (*qtd)--; // Reduz tamanho logico
            return 1;
        }
    }
    return 0;
}
```