Lista 8 – FUP – Prof. Camilo

INFORMAÇÕES

Conteúdo envolvido:

- Entrada de dados númericos
- Saída de dados númericos
- Cálculos ou transformações simples
- Uso de estrutura de seleção/condicional
- Uso de estrutura de repetição
- Especificação e uso de funções
- Uso de vetores/matrizes para armazenamento de dados
- Cadeia de caracteres (Strings)
- Ponteiros (até Questão 6)
- Alocação dinâmica de memória (Questão 7 em diante)

Material didático:

- Livro Laureano: Capítulo 11
- Apostila Manssour: Ponteiros
- Livro Laureano: Capítulo 13

OBSERVAÇÃO: PARA AS QUESTÕES QUE PEDEM "UMA FUNÇÃO" FICA IMPLÍTICO QUE UM PROGRAMA (MAIN) DEVE SER CONSTRUÍDO PARA TESTAR A FUNÇÃO.

Lista 8 – FUP – Prof. Camilo

QUESTÃO 0

Baixe, compile e execute o programa ‘manipulacao_ponteiros.c’ na pasta compartilhada. Leia o código, execute para ver o comportamento.

QUESTÃO 1

Construa uma função que recebe um vetor de números reais duplos (*double*) por referência (ponteiro) e o tamanho do vetor, e dentro da função faça a truncagem dos valores para remover os decimais (e.g. ‘12.89’ fica ‘12.0’). A **função deve alterar o vetor original**. Veja a assinatura proposta:

```
void trunca(double *vetor, int tamanho);
```

Dentro da sua função, use a função trunc() da math.h.

```
Sintaxe:

    double trunc (double x);

Parâmetros:

    x: leva um valor duplo como entrada e, em seguida, trunca os
    valores após a vírgula decimal.
```

QUESTÃO 2

Faça uma função que recebe 2 inteiros por referência (ponteiro) e **retorna o ponteiro** para o maior valor.

```
int* maiorValor(int *a, int *b);
```

QUESTÃO 3

Faça uma função que recebe um vetor de inteiros por referência, o tamanho do vetor, e duas variaveis ‘menor’ e ‘maior’ por referência. A função deve achar no vetor os valores do menor e maior números e **guardá-los nas variáveis passadas por parâmetro**.

```
void menorMaior(int *vetor, int tam, int *menor, int* maior);
```

Lista 8 – FUP – Prof. Camilo

QUESTÃO 4

Faça uma função que receba um valor inteiro N, aloque em memória um vetor de inteiros positivos de tamanho N, e preencha esse vetor com os N primeiros números múltiplos de 3. A **função deve retornar o ponteiro para o novo vetor alocado**.

```
int* primeirosMultiplosde3(int tamanho);
```

QUESTÃO 5

Faça uma função que recebe um ponteiro para vetor de inteiros positivos (e o tamanho do vetor), e que **retorna o ponteiro para o primeiro número primo da sequencia**.

```
int* primeiroPrimo(int *vetor, int tamanho);
```

QUESTÃO 6

Faça uma função que recebe um ponteiro para vetor de inteiros positivos (e o tamanho do vetor), e um ponteiro duplo (ponteiro para ponteiro) para inteiro. Identifique a posição no vetor em que está o maior número, e **guarde esse ponteiro na variável passada por referência**.

```
void maiorValor(int *vetor, int tamanho, int **p_maior);
```

Lista 8 – FUP – Prof. Camilo

QUESTÃO 7

Escreva um programa que receba do usuário um valor inteiro N, aloque espaço de memória para um vetor de inteiros com tamanho N, e gere números aleatórios entre 1 e 100 para preenchê-lo completamente. Imprima o vetor gerado.

QUESTÃO 8

Escreva um programa que receba do usuário dois valores inteiros L e C. O programa deve alocar uma matriz L x C usando o método de espaço contíguo. Gere números aleatórios entre 1 e 100 para preenchê-la completamente e imprima a matriz.

QUESTÃO 9

Escreva um programa que receba do usuário dois valores inteiros L e C. O programa deve alocar uma matriz L x C usando o método de vetor de ponteiros. Gere números aleatórios entre 1 e 100 para preenchê-la completamente e imprima a matriz.

QUESTÃO 10

Escreva uma função que recebe duas matrizes `m1` e `m2`, sendo que para cada matriz são informados também o número de linhas e colunas (`l1, c1, l2, c2`). A função deve realizar a multiplicação das matrizes, caso seja possível (`c1 == l2`), e retornar o ponteiro para a matriz resultado (`l1 x c2`). Use o método de alocação de vetor de ponteiros, e considere que as matrizes de entrada estão alocadas dessa forma.

```
int** multiplica(int** m1, int l1, int c1, int** m2, int l2, int c2);
```