
**USP – ICMC – SSC**

## SSC0501 - Introdução à Ciência da Computação I (Teórica)

**Professor responsável:** *Fernando Santos Osório*
**Semestre:** 2010/1
**Bolsista PAE:** Gustavo Pessin
**Monitor:** Matheus Lin (Seg. e Sexta 18h -19h)
**Horário:** Seg. 21h-22h40 e Terça 19h-20h40

**E-mail:** fosorio .at. icmc .dot. USP .dot. br
fosorio .at. gmail .dot. com
pessin .at. gmail .dot. com
matheus.lin .at. gmail .dot. com
**Web:** ***http://www.icmc.usp.br/~fosorio /***

# *LISTA DE EXERCÍCIOS - Nro. 09*

## [Ponteiros e Alocação Dinâmica de Memória]

1. Escreva um trecho de código em "C" para fazer a alocação dinâmica (*calloc* ou *malloc*) dos blocos de dados conforme solicitado abaixo:

   A) Vetor de 1024 Bytes (1Kbyte).
   B) Tabela de Inteiros de dimensão 10x10.
   C) Tabela para armazenar um vetor de 50 registros contendo: nome do produto (30 caracteres), código do produto (inteiro) e preço em reais.
   D) Texto de até 100 linhas com até 80 caracteres em cada linha.

2. Faça um programa para armazenar em memória um vetor de dados contendo 1500 valores do tipo int, usando a função de alocação dinâmica de memória CALLOC:

   1a) Faça um loop e verifique se o vetor contém realmente os 1500 valores inicializados com zero (conte os 1500 zeros do vetor).
   1b) Atribua para cada elemento do vetor o valor do seu índice junto a este vetor.
   1c) Faça um loop e conte quantos dos 1500 valores são diferente de zero no vetor. Exiba na tela o valor final da contagem.
   1d) Exibir na tela os 10 primeiros e os 10 últimos elementos do vetor.

3. Faça um programa que pergunte ao usuário quantos valores ele deseja armazenar em um vetor de doubles, depois use a função MALLOC para reservar (alocar) o espaço de memória de acordo com o especificado pelo usuário. Use este vetor dinâmico como um vetor comum, atribuindo aos 10 primeiros elementos do vetor valores aleatórios (rand) entre 0 e 100. Exiba na tela os valores armazenados nos 10 primeiros elementos do vetor (O vetor deve ter pelo menos um tamanho igual a 10 doubles, ou mais).

4. Faça um laço de entrada de dados, onde o usuário deve digitar uma seqüência de números, sem limite de quantidade de dados a ser fornecida. O usuário irá digitar os números um a um, sendo que caso ele deseje encerrar a entrada de dados, ele ira digitar o número Zero. No final, todos os dados digitados deverão ser salvos em um arquivo texto em disco. *Atenção*: os dados devem ser armazenados na memória deste modo... faça com que o programa inicie criando um ponteiro para um bloco (vetor) de 10 valores inteiros, e alocando dinamicamente espaço em memória para este bloco; após, caso o vetor alocado esteja cheio; aloque um novo vetor do tamanho do vetor anterior adicionado com espaço para mais 10 valores (tamanho N+10, onde N inicia com 10), copie os valores já digitados da área inicial para esta área maior e libere a memória da área inicial; repita este procedimento de expandir dinamicamente com mais 10 valores o vetor alocado cada vez que o mesmo estiver cheio. Assim o vetor irá ser "expandido" de 10 em 10 valores.

5. Faça um programa para ler dados do disco, adicionar dados e salvar em disco os dados que descrevem o cadastro de produtos de uma loja, contendo as seguintes informações: código do produto (até 5 dígitos), nome/descrição do produto (79 caracteres+'\0' = 80), e preço do produto (em reais). O programa deve declarar um registro (typedef/struct) para agrupar os dados, que serão lidos do disco (arquivo "entrada.txt"), armazenados em memória em um vetor com alocação dinâmica e depois gravados novamente em disco (arquivo "saida.txt"). O arquivo em disco é um arquivo do tipo texto, com um dado em cada linha, sendo que a primeira linha do arquivo indica quantos registros estão gravados neste arquivo. Uma vez lido o arquivo, perguntar para o usuário quantos dados ele deseja adicionar ao cadastro, criar um vetor com alocação dinâmica, e armazenar os dados digitados neste vetor. Por fim, salvar em disco os dados lidos do arquivo juntamente com os novos dados digitados pelo usuário. O novo arquivo também deve iniciar com uma primeira linha indicando o total de produtos cadastrados (os antigos mais os novos), sendo gravado em formato texto com um dado em cada linha. Exemplo:

```
>> Cadastro de Produtos da Loja ACME <<

** Lendo Arquivo **
Total de registros no arquivo: 30

** Entrada de Dados **

Quantos registros novos voce deseja criar? 10

Produto 1 - Codigo:  12345
Produto 1 - Descricao:  Caixa_de_Bits_de_Paridade_Avulsos_(10_unidades)
Produto 1 - Preço:  1.99

Produto 2 - Codigo:  98765
Produto 2 - Descricao:  Memória_WROM_Quântico-Neural_de_100_Terabytes
Produto 2 - Preço:  100.00
...
Produto 10 - Codigo:  10101
Produto 10 - Descricao:  Monitor_Holístico-Holográfico_1-D
Produto 10 - Preço:  99.99

** Gravando Dados **
FIM!
```

6. Faça um programa que simule "virtualmente" a memória de um computador: o usuário começa especificando o tamanho da memória (define quantos bytes tem a memória), e depois ele irá ter 2 opções: inserir um dado em um determinado endereço, ou consultar o dado contido em um determinado endereço. A memória deve iniciar com todos os dados zerados.

7. Baseado no programa anterior, implemente um mecanismo para associar nomes as posições de memória (um nome de uma posição de memória tem até 10 caracteres, com o '\0'). O usuário irá poder usar 5 opções diferentes para manipular a memória: 1) Associar um nome com uma posição de memória; 2) Informar um endereço e um valor para armazenar neste endereço; 3) Informar um nome de uma posição de memória e armazenar um valor nesta posição; 4) Pedir para recuperar o dado contido em uma posição de memória; 5) Pedir para recuperar o dado, indicando o nome da posição de memória onde ele se encontra.

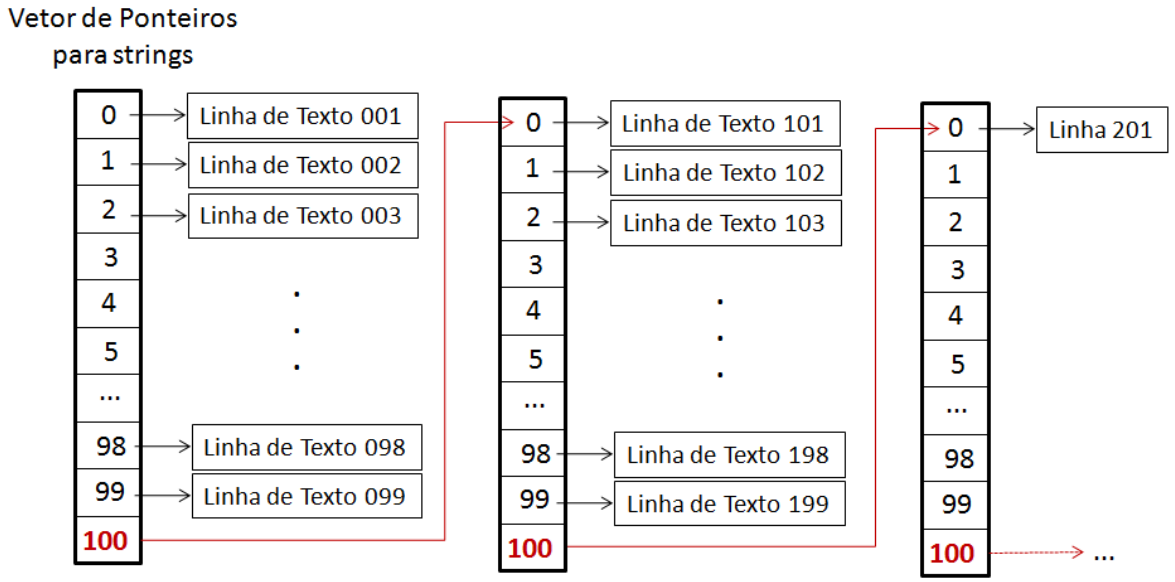
8. PERGUNTA: Responda e explique claramente suas respostas as questões que seguem abaixo.
Podemos salvar ponteiros em disco? Faz sentido?

9. TESTE DE MESA:
Faça um teste de mesa para o programa abaixo (sem uso do computador) e indique qual será o valor das variáveis escritas na tela nos comandos *printf* indicados no código abaixo

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{

  int a,b,c;
  int  vetor[3];

  int *pti;
  int *pti1, *pti2;

  a=10;
  b=20;
  c=30;

  pti1  = &a;
  pti2  = &b;
  *pti1 = c;
  *pti2 = c;

  pti  = vetor;
  *pti = 1;
  *(pti+1) = 2;
  *(pti+2) = 3;

  printf (">> Teste de Mesa (1) << \n");
  printf ("A: %d, B: %d, C: %d\n",a,b,c);
  printf ("Vetor: %d, %d, %d\n",vetor[0],vetor[1],vetor[2]);

  a=99;
  vetor[0]  = *pti1;
  vetor[1]  = *pti2;
  *(vetor+2)= *pti;

  printf (">> Teste de Mesa (2) << \n");
  printf ("A: %d, B: %d, C: %d\n",a,b,c);
  printf ("Vetor: %d, %d, %d\n",vetor[0],vetor[1],vetor[2]);

  system("PAUSE");
  return 0;

}
```

**Exercícios** – *QUESTÃO DESAFIO*:

10. Questão Desafio: Faça uma editor de textos, onde o texto possa ter tantas linhas quanto o usuário desejar. Use uma estrutura de dados em memória composta por um vetor de ponteiros para strings, ou seja, cada elemento do vetor é um ponteiro que aponta para uma cadeia de caracteres. Cada nova linha de texto deve ser alocada dinamicamente com o comando calloc/malloc, e o ponteiro para esta linha de texto deve ser adicionado ao vetor de ponteiros. O vetor de ponteiros deve inicialmente prever um certo tamanho, por exemplo, 100 ponteiros (texto de 100 linhas), e quando as 100 linhas estiverem ocupadas, você deve expandir este vetor de ponteiros. Para expandir o vetor de ponteiros adote a seguinte estratégia: o último ponteiro do vetor, ao invés de conter o endereço de uma string irá conter o endereço de mais uma tabela de 100 ponteiros, alocada dinamicamente. A figura abaixo apresenta um esquema de como ficará a estrutura de dados prevista para este editor de textos.




---

Lista complementares de exercícios disponível junto a Wiki ICMC da Disciplina SSC0501
http://wiki.icmc.usp.br/index.php/SSC-501 - Listas de Exercícios

---