Universidade Estadual de Campinas
Algoritmos e Programação de Computadores (MC102 Turma Z)

Lista de Exercícios 10

1. Suponha que os endereços das variáveis a, b e c são 1000, 1004 e 1008, respectivamente.

```
float a, b;
float c, *pa, *pb;
a = 0.001;
b = 0.003;
pa = &a;
*pa = 2 * a;
pb = &b;
c = 3 * (*pa + *pb);
```

   Após a execução do trecho de código acima, quais os valores de:
   (a) pa (b) *pa (c) pb (d) c

2. Escreva um procedimento em linguagem C que receba um ponteiro para um vetor de inteiros e o número de inteiros no vetor por parâmetro, e incremente apenas os valores contidos em índices pares do vetor. Escreva o acesso ao vetor de duas formas: utilizando colchetes e índices, como em vetores, e evitando-os.

3. Escreva um algoritmo que receba um conjunto de inteiros do usuário, não sabendo a princípio quantos inteiros serão digitados (o usuário digitará zero para finalizar a entrada). O seu programa deverá utilizar o mínimo possível de memória, e deverá guardar todos os inteiros digitados em um vetor, imprimindo o conteúdo do vetor em seguida. Existe alguma maneira mais eficiente (leia-se que não necessite de um grande número de chamadas à alocação de memória dinâmica) de realizar a tarefa?

4. Modifique o algoritmo escrito a partir da questão 2 da lista 9 de forma que o valor total a pagar seja não apenas impresso, mas guardado num novo campo presente no registro do participante do evento. Reescreva este algoritmo de forma que haja um procedimento que realize o cálculo e este procedimento deve receber um ponteiro para o registro de apenas um participante por vez.

5. Considere a declaração de um ponteiro para inteiro chamado p. Indique o que cada um dos comandos a seguir significa:
   a. p++;
   b. (*p)++;
   c. *(p++);
   d. *(p+10) = 5;

6. Qual o valor escrito ao final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. Indique o valor de cada uma das variáveis a cada atribuição realizada.

```
int main() {
  int y, *p, x;
  y = 0;
  p = &y;
  x = *p;
  x = 4;
  (*p)++;
```

```
    x--;
    (*p) += x;
    printf ("y = %d\n", y);
    return(0);
}
```

7. 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).

8. Escreva um programa que declare uma matriz *n*x*n* de inteiros (lendo as dimensões do teclado e alocando a menor quantidade de memória possível para armazenar os valores da matriz). Você deve inicializar a matriz com zeros usando ponteiros para endereçar seus elementos. Preencha depois a matriz com os números de 1 a 10000, também usando ponteiros.

9. Faça um programa que multiplique duas matrizes. O programa deverá estar estruturado de maneira que:
   a. O usuário forneça as dimensões das matrizes (teste se as dimensões são compatíveis, isto é, se as matrizes podem ser multiplicadas);
   b. As matrizes sejam alocadas dinamicamente (faça uma função para isto);
   c. As matrizes sejam lidas pelo teclado (faça uma função para leitura das matrizes de qualquer dimensão);
   d. As matrizes sejam multiplicadas (faça uma função para a multiplicação);
   e. A matriz resultante seja apresentada na tela (faça uma função para apresentar a matriz na tela).

10. 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.

11. Faça um programa que simule 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.

12. Baseado no programa anterior, implemente um mecanismo para associar nomes às 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.