**MAC2166 – Introdução à Computação**
Escola Politécnica
Terceira Prova – 23 de junho de 2015
**(Gabarito)**

## Questão 1 (valor: 3 pontos)

Escreva uma função em linguagem C de protótipo

```
int testa_solucao(int m, int n, double A[][MAX], double *b, double *x, double eps);
```

que recebe inteiros $m$ e $n$, uma matriz $A$ com $m$ linhas e $n$ colunas, um vetor $b$ com $m$ elementos, um vetor $x$ com $n$ elementos e um número $eps > 0$, e testa se $Ax = b$. Mais precisamente, sua função deve devolver um número NÃO-ZERO se para todo $i = 0, \ldots, m-1$ temos

$$|(Ax - b)_i| \leq eps.$$

Caso contrário, sua função deve devolver ZERO.

**Solução**

```
int testa_solucao(int m, int n, double A[MAX][MAX], double *b, double *x, double eps)
{
    int i,j, eh_solucao = 1;
    double soma;

    for (i = 0; eh_solucao && i < m; i++)
    {
        soma = 0;
        for (j = 0; j < n; j++)
            soma += A[i][j] * x[j];

        if (soma-b[i] > eps || soma-b[i] < -eps)
            eh_solucao = 0;
    }

    return eh_solucao;
}
```

## Questão 2 (valor: 3 pontos)

Escreva uma função em linguagem C de protótipo

```
void converte2(int n, char *s);
```

que recebe um inteiro $n$ (que pode ser positivo ou negativo!) e um ponteiro para um vetor de caracteres $s$, e coloca em $s$ a sequência de caracteres (string) que é a representação do número $n$ na base 2. Por exemplo, se $n = -22$, devemos ter ao final da chamada $s =$ `"-10110"`. LEMBRE-SE: sequências de caracteres terminam com o caracter nulo!

Em sua implementação você NÃO precisa se preocupar com quantos caracteres o vetor $s$ pode comportar. **ATENÇÃO:** Nesta questão você **NÃO** pode usar as funções declaradas em `string.h`.

**BÔNUS (1 PONTO EXTRA).** Em vez de fazer a função acima, escreva uma função de protótipo

```
char *converte2(int n);
```

que devolve uma sequência de caracteres contendo a representação binária do número $n$. A sequência de caracteres deve ser alocada dinamicamente na memória pela sua função através da função *malloc*.

**Solução 1 – sem alocação dinâmica de memória**

```
void converte2(int n, char *s)
{
    char digito;
    int cont = 0, ncopia;

    if (n == 0)
    {
        s[0] = '0';
        cont++;
    }
    else if (n < 0)
    {  /* Se o n e' negativo, coloca na primeira posicao da string o caracter de sinal */
        s[0] = '-';
        n = -n;
        cont++;
    }

    ncopia = n;
    while (ncopia > 0)   /* conta o numero de digitos de n na notacao binaria */
    {
        cont++;
        ncopia /= 2;
    }

    s[cont] = 0;        /* Inclui o caracter nulo no final da string */

    /* Obtem os digitos do numero n na notacao binaria e o armazena em sua posicao correta em s */
    while (n > 0)
    {
        cont--;
        digito = '0' + (n % 2);
        s[cont] = digito;
        n /= 2;
    }
}
```

**Solução 2 – com alocação dinâmica de memória**

```
#include <stdlib.h>

/* Conta o numero de digitos de n na notacao binaria */
int conta_digitos_binarios(int n)
{
    int cont = 0;

    if (n == 0)
        return 1;

    if (n < 0)
    {   /* Se o n e' negativo, coloca na primeira posicao da string o caracter de sinal */
        cont++;
        n = -n;
    }

    /* Obtem os digitos do numero n na notacao binaria e o armazena em sua posicao correta em s */
    while (n > 0)
    {
        cont++;
        n /= 2;
    }

    return cont;
}

char *converte2(int n)
{
    char digito, *s;
    int cont;

    cont = conta_digitos_binarios(n);

    /* aloca dinamicamente espaco para a string com a representacao binaria de n */
    s = (char *) malloc ((cont + 1) * sizeof(char));

    if (n == 0)
        s[0] = '0';
    else if (n < 0)
    {
        s[0] = '-';
        n = -n;
    }

    s[cont] = 0;         /* Inclui o caracter nulo no final da string */

    /* Obtem os digitos do numero n na notacao binaria e o armazena em sua posicao correta em s */
    while (n > 0)
    {
        cont--;
        digito = '0' + (n % 2);
        s[cont] = digito;
        n /= 2;
    }

    return s;
}
```

## Questão 3 (valor: 4 pontos)

Uma matriz real $A$ de $m+1$ linhas e $n+1$ colunas representa o polinômio de duas variáveis

$$p(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} A_{ij} x^i y^j.$$

Por exemplo, a matriz

$$\begin{pmatrix} 1 & 2 & 3 \\ -1 & 0 & 4 \end{pmatrix}$$

representa o polinômio

$$p(x,y) = 1 + 2y + 3y^2 - x + 4xy^2.$$

Escreva um programa em linguagem C que

- Lê um polinômio. Para isso primeiro lê números inteiros $m$, $n$ e $k$ digitados pelo usuário e em seguida lê $k$ triplas de números $(i, j, c)$, onde $i$, $j$ são inteiros tais que $0 \leq i \leq m$ e $0 \leq j \leq n$ e $c$ é um número real. A tripla $(i, j, c)$ representa o monômio $cx^iy^j$; juntas as triplas fornecidas especificam os termos não nulos de um polinômio de duas variáveis.
- Calcule seu polinômio para um conjunto de valores de x e y. Seu programa então deve ler um número inteiro $N$ e em seguida $N$ pontos $(x, y)$ e para cada ponto escrever na saída o valor do polinômio calculado no¡ ponto.

Você deve supor que já existe uma função de protótipo

```
double potencia(double x, double y);
```

que devolve o valor de $x^y$ escrita no arquivo `potencia.c`, que está no mesmo diretório do seu programa, e que deve ser incluído em seu programa.

Para armazenar os coeficientes do polinômio você deve **obrigatoriamente** utilizar uma matriz.

**BÔNUS (1 PONTO EXTRA).** Use alocação dinâmica de memória para guardar a matriz $(m+1) \times (n+1)$ com os coeficientes do polinômio.

**Solução 1 – sem alocação dinâmica de memória**

```
#include <stdio.h>
#include <math.h>
#define MAX 100

#include "potencia.c"

double calcula_polinomio(double coef[][MAX], int m, int n, double x, double y)
{
    double soma = 0;
    int i, j;

    for (i = 0; i <= m; i++)
        for (j = 0; j <= n; j++)
            soma += coef[i][j] * potencia(x,i) * potencia(y,j);

    return soma;
}

int main()
{
    int m, n, k, N, i, j;
    double x, y, coef[MAX][MAX];

    printf("Digite os valores de m, n e k (num. inteiros): ");
    scanf("%d %d %d", &m, &n, &k);

    /* Zera a matriz de coeficientes do polimonio */
    for (i = 0; i <= m; i++)
        for (j = 0; j <= n; j++)
            coef[i][j] = 0;

    /* Le os valores dos coeficientes nao nulos do polinomio */
    while (k > 0)
    {
        printf("Digite uma tripla de valores com i, j (num. inteiros) e c (num real): ");
        scanf("%d %d", &i, &j);
        scanf("%lf", &coef[i][j]);
        k--;
    }

    printf("Digite o valor de N: ");
    scanf("%d", &N);

    /* Le N pontos (x,y) e mostra o valor do polinomio para cada um deles */
    while (N > 0)
    {
        printf("Digite um par de valores (num. reais) para x e y: ");
        scanf("%lf %lf", &x, &y);
        printf("Valor do polinomio: %f", calcula_polinomio(coef, m, n, x, y));
        N--;
    }

    return 0;
}
```

**Solução 2 – com alocação dinâmica de memória**

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

#include "potencia.c"

/* Funcao que aloca dinamicamente espaco na memoria para uma
   matriz de tamanho m x n e devolve um ponteiro para a matriz */
double **aloca_matriz(int m, int n)
{
    double **mat;
    int i;

    /* Uma matriz pode ser vista como um vetor de linhas.
       Cada linha, por sua vez, tambem e' um vetor.
       Assim, precisamos alocar espaco para m+1 vetores */

    /* Aloca espaco para o vetor de linhas */
    mat = malloc(m * sizeof (double *));

    /* Aloca espaco para cada linha individualmente */
    for (i = 0; i < m; ++i)
        mat[i] = (double *) malloc(n * sizeof (double));

    return mat;
}

double calcula_polinomio(double **coef, int m, int n, double x, double y)
{
    double soma = 0;
    int i, j;

    for (i = 0; i <= m; i++)
        for (j = 0; j <= n; j++)
            soma += coef[i][j] * potencia(x,i) * potencia(y,j);

    return soma;
}

int main()
{
    int m, n, k, N, i, j;
    double x, y, **coef;

    printf("Digite os valores de m, n e k (num. inteiros): ");
    scanf("%d %d %d", &m, &n, &k);

    /* Aloca dinamicamente uma matriz de tamanho mxn,
       para os coeficientes do polinomio */
    coef = aloca_matriz(m+1,n+1);

    /* Zera a matriz de coeficientes do polimonio */
    for (i = 0; i <= m; i++)
        for (j = 0; j <= n; j++)
            coef[i][j] = 0;
```

```
    /* Le os valores dos coeficientes nao nulos do polinomio */
    while (k > 0)
    {
        printf("Digite uma tripla de valores com i, j (num. inteiros) e c (num real): ");
        scanf("%d %d", &i, &j);
        scanf("%lf", &coef[i][j]);
        k--;
    }

    printf("Digite o valor de N: ");
    scanf("%d", &N);

    /* Le N pontos (x,y) e mostra o valor do polinomio para cada um deles */
    while (N > 0)
    {
        printf("Digite um par de valores (num. reais) para x e y: ");
        scanf("%lf %lf", &x, &y);
        printf("Valor do polinomio: %f", calcula_polinomio(coef, m, n, x, y));
        N--;
    }

    /* Nao precisa desalocar o matriz coef, pois o programa ja vai ser encerrado */

    return 0;
}
```