INTRODUÇÃO À PROGRAMAÇÃO

3ª Prova - 09/12/2010 - Prof. Marcus Ramos

1. (2.5 pontos) Faça um programa que manipule uma matriz bidimensional de números inteiros da seguinte forma:

    a. Inicialmente a matriz é preenchida com valores digitados pelo usuário;
    b. O programa elimina a primeira linha e a primeira coluna, mantendo os demais valores;
    c. O programa insere uma linha contendo apenas zeros no final (embaixo) e uma coluna contendo apenas zeros no final (à direita);
    d. O programa imprime o conteúdo da nova matriz na tela.

    Deve-se usar o comando `define` para especificar a quantidade de linhas e de colunas da matriz. Exemplo de entrada e respectiva saída:

1	2	3	4	6	7	8	0
5	6	7	8	10	11	12	0
9	10	11	12	14	15	16	0
13	14	15	16	0	0	0	0

```
#define lin 4
#define col 4
int main () {
  int m[lin][col];
  int i,j;
  for (i=0;i<lin;i++)
       for (j=0;j<col;j++)
            scanf ("%d",&m[i][j]);
  for (j=0;j<(col-1);j++)
       for (i=0;i<(lin-1);i++)
            m[i][j]=m[i+1][j+1];
  for (i=0;i<lin;i++) m[i][col-1]=0;
  for (j=0;j<col;j++) m[lin-1][j]=0;
  for (i=0;i<lin;i++) {
       for (j=0;j<col;j++)
            printf ("%d\t",m[i][j]);
       printf ("\n");
       }
  }
```

2. (2.5 pontos) Faça uma função que aceite como parâmetro um valor em ponto flutante x e um número inteiro n, e retorne como resultado e^x, calculado através da fórmula de Taylor:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, \qquad -\infty < x < \infty$$

    O número inteiro n ≥ 1 informa a quantidade de termos que deverão ser usados no cálculo de e^x através da fórmula acima. Ilustre o uso da função em um programa que calcula e^x, com x e n informados pelo usuário.

Exemplos:

x=1 e n=6

$$e^1 = 1 + \frac{1}{1!} + \frac{1^2}{2!} + \frac{1^3}{3!} + \frac{1^4}{4!} + \frac{1^5}{5!} = 2.666667$$

x=2 e n=11

$$e^2 = 1 + \frac{2}{1!} + \frac{2^2}{2!} + \frac{2^3}{3!} + \cdots + \frac{2^{10}}{10!} = 7.387302$$

```
#include <math.h>
#include <stdio.h>
int fatorial (int n) {
  if (n==0) return 1;
       else return n*fatorial(n-1);
  }
float taylor (float x, int n) {
  int i;
  float soma=0;
  for (i=1;i<n;i++)
       soma=soma+pow(x,(i-1))/fatorial(i-1);
  return soma;
  }
int main () {
  float x;
  int n;
  scanf ("%f",&x);
  scanf ("%d",&n);
  printf ("\n%f\n",taylor(x,n));
  }
```

3. (2.5 pontos) Faça um programa que leia uma cadeia de até 100 caracteres e efetue as seguintes manipulações:

   a. Substitua todas as ocorrências da seqüência "abc" pela seqüência "ABC";
   b. Substitua todas as ocorrências da seqüência "xyz" pela seqüência "-n-", onde n representa o número de ordem de cada ocorrência "xyz" na cadeia de entrada (considerar n ≤ 9);
   c. Imprima a cadeia resultante na tela.

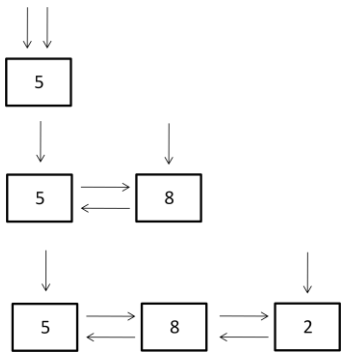
Exemplo de entrada: amabcjhxyzdergabcklpxyzvm
Exemplo de saída: amABCjh-1-dergABCklp-2-vm

```
#include <stdio.h>
#include <string.h>
int main () {
  char s[101];
  char *p;
  char cont='1';
  gets (s);
  p=strstr (s,"abc");
  while (p) {
       *p='A';
```

```
        p++;
        *p='B';
        p++;
        *p='C';
        p=strstr (s,"abc");
        }
  p=strstr (s,"xyz");
  while (p) {
        *p='-';
        p++;
        *p=cont;
        cont++;
        p++;
        *p='-';
        p=strstr (s,"xyz");
        }
  puts (s);
}
```

4. (2.5 pontos) Faça um programa que monte uma lista duplamente ligada de registros contendo números inteiros digitados pelo usuário, usando alocação dinâmica. Manter um ponteiro para o início da lista e outro para o final. Novos números digitados pelo usuário devem sempre ser inseridos no final da lista. O programa encerra quando o usuário digitar o valor zero. Quando isso acontecer, o programa deve imprimir a ordem direta dos números digitados (usando para isso o ponteiro de início), e também a ordem inversa (usando o ponteiro de final).

Exemplo:




```
#include <stdlib.h>
#include <stdio.h>
struct item {
  int n;
  struct item *p;
  struct item *a;
  };
int main () {
  struct item *i=NULL, *f=NULL, *aux;
  int n;
  scanf ("%d",&n);
  while (n) {
        if (i) {
                    aux=malloc (sizeof (struct item));
                    f->p=aux;
                    aux->n=n;
```

```
                aux->p=NULL;
                aux->a=f;
                f=aux;
                }
          else {
                      i=malloc (sizeof (struct item));
                      i->n=n;
                      i->p=NULL;
                      i->a=NULL;
                      f=i;
                      }
     scanf ("%d",&n);
     }
printf ("Ordem direta:\n");
while (i) {
                printf ("%d\t",i->n);
                i=i->p;
                }
printf ("\nOrdem inversa:\n");
while (f) {
                printf ("%d\t",f->n);
                f=f->a;
                }
}
```