Computação para Informática - Prof. Adriano Joaquim de Oliveira Cruz

O objetivo desta aula prática é exercitar funções sem ponteiros e sem recursão.

Exercício 1: Complete o programa mostrado na listagem 1. Este programa lê uma série de números inteiros e calcula e imprime os fatoriais. A série termina quando um número negativo é lido.

Listing 1: Programa do problema 1.

```
#include<stdio.h>
long int fat (long int n);
int main (void) {
     long int numero;
     while (1)
     {
          scanf("%ld", &numero);
          if (numero < 0) break;
          printf("O fatorial de %ld vale %ld\n",
               numero, fat(numero));
     }
     return 0;
}
long int fat (long int n) {
}
```

Exercício 2: Complete o programa mostrado na listagem 2. Este programa calcula a combinação de n elementos tomados p a p, que é dada pela fórmula

$$C_n^p = \frac{n!}{(n-p)! \times p!}$$

Por exemplo, com esta fórmula podemos calcular de quantas maneiras diferentes podemos juntar 10 cartas de um baralho com 40, ou C_{40}^{10}.

Listing 2: Programa do problema 2.

```
#include<stdio.h>

long int fat(long int );
long int c (long int n, long int p);

int main (void) {
     long int n, p;
     while (1) {
          scanf("%ld %ld", &n, &p);
          if (n < p) break;
          printf("O valor da combinacao de %ld %ld a %ld vale %ld\n",
               n, p, p, c(n, p));
     }
     return 0;
}
long int fat (long int n) {
}
long int c (long int n, long int p) {
}
```

Exercício 3: Considere a função

`int QuantasVezes (char frase[], char procurado);`

Esta função retorna quantas vezes o caracter (`procurado`) apareceu em uma cadeia de caracteres `frase`. O trecho de programa abaixo imprime quantas vezes apareceu em uma frase um determinado caracter.

```
char c;
char frase[80];
int vezes;
/* ... */

gets(frase);
c = 'a';
vezes = QuantasVezes(frase, c);
printf("O caracter %c apareceu %d vezez\n", c, vezes);

/* ... */
```

Escreva a função `QuantasVezes` e um programa que, usando esta função imprima a freqüência das vogais em uma frase.

Exercício 4: Escreva um programa que leia uma frase de até 80 caracteres e a imprima após alterá-la segundo a criptografia de Cesar. Neste método cada letra da frase passa a ser letra seguinte do alfabeto. A letra 'Z' vira a letra 'A'.

Importante: Somente letras podem ser modificadas. Letras maiúsculas continuam maiúsculas e e o mesmo acontece com letras minúsculas.

Neste exercício você deve criar a função **void** `criptografa (char frase[])`.

Exercício 5: Escreva a função **void** `decriptgrafa(char frase[])` que recebe uma frase criptografada segundo o algoritmo mostrado no exercício anterior e decriptgrafa. Escreva um programa que usa esta função.

Exercício 6: Considere que você está controlando um robo que se move em um mundo plano de largura e altura definidas (Figura 1). A sua tarefa é completar o programa mostrado na listagem 3. Este programa lê os comandos do teclado e altera as posições do robo. Observe que se o você emitir uma ordem inválida o robo não se move. Por exemplo, veja na listagem que se ele está no limite direito do mundo e você deu uma ordem para ele andar para a direita, neste caso o robo fica parado.

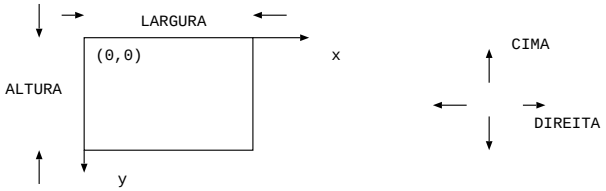

Figura 1: Mundo Plano

Listing 3: Programa do problema 7.

```
#include<stdio.h>

#define ALTURA 10
#define LARGURA 10

int MoveEsquerda(int x);
int MoveDireita(int x);
int MoveCima(int y);
int MoveBaixo(int y);

int main (void) {
     int px = py = 0;
     char ordem;
     int continua = 1;

     while (continua) {
          ordem = getchar();
          switch(ordem) {
               case 'a': case 'A':
                    px = MoveEsquerda(px); break;
               case 'd': case 'D':
                    px = MoveDireita(px); break;
               case 'w': case 'W':
                    py = MoveCima(py); break;
               case 'x': case 'X':
                    py = MoveBaixo(py); break;
               case 'q': case 'Q':
                    continua = 0; break;
               default:
                    puts("Opcao invalida");
          }
          printf("Estou em %d %d\n", px, py);
     }
     return 0;
}

int MoveEsquerda(int x) {
}
int MoveDireita(int x) {
     if (x < LARGURA) x++;
     return x;
}
int MoveCima(int y) {
}
int MoveBaixo(int y) {
}
```

Exercício 7: Uma das maneiras de calcular o valor de π é usar a série

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} \cdots$$

Escreva um programa que leia o número de termos da série que usuário deseja usar para calcular π e imprima o valor calculado. Para calcular o valor defina uma função que tenha o seguinte protótipo:
`double Pi (int termos);`

Exercício 8: Complete as partes que faltam no programa 4. Os pedaços que devem ser completados estão indicados por */* Aqui falta codigo*/*.

Listing 4: Programa do problema 8.

```
#include<stdio.h>

#define TAM 81
void ConverteCadeiaParaMaiusculas(char v[]);
int TamanhoCadeia (char v[]);
char ConverteLetraParaMaiuscula(char c);
int EhLetraMinuscula (char c);

int main (void) {
     char frase[TAM];

     gets(frase);
     while (TamanhoCadeia(frase)) {
          ConverteCadeiaParaMaiusculas(frase);
          puts(frase);
          gets(frase);
     }
     return 0;
}
void ConverteCadeiaParaMaiusculas(char v[]) {
     int i;

     for (i=0; i<TamanhoCadeia(v); i++) {
          v[i] = ConverteLetraParaMaiuscula(v[i]);
     }
}
char ConverteLetraParaMaiuscula(char c) {
     if (EhLetraMinuscula(c)) {
          /* aqui falta codigo */
     }
     return c;
}
int EhLetraMinuscula (char c) {
     /* aqui falta codigo */
}
int TamanhoCadeia (char v[]) {
     /* Aqui falta codigo */
}
```

Exercício 9: Complete as partes que faltam no programa 5. Os pedaços que devem ser completados estão indicados por */* Aqui falta codigo*/*.

Listing 5: Programa do problema 9.

```
#include<stdio.h>

#define TAM 13571113

double MaiorElementoVetor(double vetor[], int tam);
double MediaVetor(double vetor[], int tam);
void OrdenaVetor(double vetor[], int tam);
void ImprimeVetor(double vetor[], int tam);

int main (void)
{
```

```
    double vetor[TAM];
    double media, maior;

    LeVetor(vetor, TAM);
    maior = MaiorElementoVetor(vetor, TAM);
    media = MediaVetor(vetor, TAM):
    printf("Maior elemento e: %f\n", maior);
    printf("Media do Vetor e: %f\n", media);
    Ordena(vetor, TAM);
    ImprimeVetor, TAM):

    return 0;
}
double MaiorElementoVetor(double vetor[], int tam) {
    /* Falta codigo */
}
double MediaVetor(double vetor[], int tam) {
    /* Falta codigo */
}
void OrdenaVetor(double vetor[], int tam) {
    /* Falta codigo */
}
void ImprimeVetor(double vetor[], int tam) {
    /* Falta codigo */
}
```

Exercício 10: Considere a função

`int ondeEsta (char frase1[], char frase2[]);`

Esta função retorna em que posição do vetor `frase1` se encontra o vetor `frase2`. Caso o vetor não apareça a função retorna o valor `-1`.

Escreva a função e um programa que, usando esta função imprima se um vetor está contido no outro, e no caso positivo em que posição ele está.

Exercício 11: Escreva um programa que converta um número inteiro positivo da base 10 para a base 2. Para isto escreva uma função que tenha o seguinte protótipo:

`void Converter (int numeroBase10, int numeroBase2[32]);`

A função recebe o número na variável `numeroBase10` e retorna todos os 32 bits no vetor `numeroBase2`. O número na base 2 deve ser armazenado no vetor da seguinte maneira: bit 31 na posição 31, bit 30 na posição 30 e assim sucessivamente.

O seu programa deve imprimir o número na ordem correta com todos os 32 dígitos.