São José dos Campos, 05 de Junho de 2008 — Instituto Tecnológico de Aeronáutica
Disciplina: CES – 10 Introdução à Computação. — Divisão de Ciência da Computação
Semestre 2008 - 2º Período
Professor: Carlos Henrique Quartucci Forster — 1º Ano de Engenharia Fundamental
Estagiária: Michelle de Oliveira Parreira — Turma 2

LISTA DE EXERCÍCIOS – MÊS 04

BACKGROUND

Passagem de Parâmetros para uma Função (VALOR x REFERÊNCIA)

Por Valor:

Na passagem por valor (como foi visto até agora), os parâmetros de uma função funcionam como variáveis suas, ou seja NADA tem a ver com as variáveis da função que a chamou. Desta forma alterações nos valores desses parâmetros não interferem nos valores das variáveis da função chamadora.

```
int soma (int x1, int x2)
{
	x1+= x2;
	return x1;
}
void main (void)
{
	int v1, v2;
	scanf("%d %d",&v1,&v2);
	printf("%d",soma(v2,v1));
}
```

Nesta função por exemplo x1 é alterado mas a variável v2 da main não é alterada. Isso ocorre porque o parâmetro x1 funciona como uma variável da função soma apenas inicializada com o **valor** de v2 (por isso se chama passagem de parâmetro por valor).

Por Referência:

No entanto pode ser interessante que uma variável na função chamadora possa ser alterada (ex: função que troque o valor de duas variáveis). Ou mesmo haja a necessidade de se retornar mais de um valor da função (pois com o *return* só é possível retornar um único valor), por exemplo função que retorne as horas e minutos dados apenas os minutos.

Para esses casos, algumas linguagens permitem a passagem de parâmetros por referência, onde uma variável passada como parâmetro ao ser alterada dentro da função tem seu valor alterado também na função chamadora (é na verdade uma única variável usada por ambas as funções).

Na linguagem C na teoria existe apenas passagem de parâmetros por valor, mas através do uso de ponteiros se consegue na prática a passagem de parâmetros por referência.

Ex:

```
void troca(int *x, int *y)
{
	int aux;
	aux=*x;
	*x=*y;
	*y=aux;
}
```

```
void main(void)
{
	int a=10,b=20;
	troca(&a,&b);
	printf("%d %d",a,b);
}
```

A presença do * na frente do nome dos parâmetros indica que esses são na verdade ponteiros. Ao passar o parâmetro por referência a função chamadora deve passar o endereço da sua variável, para isso usa-se o &. O *scanf* é um exemplo de função que altera o valor da variável passada como referência.

Observações:

- Para que seja feita a passagem por referência é imprescindível que tanto a função a ser chamada esteja recebendo em um ponteiro e a função chamadora esteja passando o endereço de uma variável.
- Como fica quando uma função chama a outra, ambas usando parâmetro passado por referência? (um exemplo de duas funções quaisquer e outro com *scanf)*
- Protótipo omitindo o nome da variável não deve omitir o *.
- Cuidado com operações aritméticas que envolvam ponteiros (ex: `(*x)++;`)

EXERCÍCIOS

1. (Variáveis locais e globais.) O que é impresso na tela pelo seguinte programa?

```
#include <stdio.h>
int w=1,x=2,y=3,z=4;
void foo(int y)
{
      int z=6;
      extern int w;
      printf("%d %d %d %d ", w,x,y,z);
}
main()
{
       int x=7;
       foo(5);
       foo(x);
       printf("%d %d %d %d.", w,x,y,z);
}
```

2. (Variáveis locais e globais.) O que é impresso na tela pelo seguinte programa?

```
#include <stdio.h>
int w=1,x=2,y=3,z=4;
void foo(int y)
{
     static int z=6;
     printf("%d %d %d %d", w,x,y,z);
     z=y;
}
main()
{
      int x=7, w=8;
      foo(5);
     {
       extern int w;
        foo(w);
      }
      printf("%d %d %d %d.", w,x,y,z);
}
```

3. (Variáveis locais e globais.) O que é impresso pelo seguinte programa?

```
#include <stdio.h>
       int x=0;
void main()
{
       int i, x=1;
       printf("%d ",x);
       for(i=0; i<4; i++)
       {
              int x=2;
```

```
                printf("%d ",x);
                {
                        int x=3;
                        printf("%d ",x);
                }
        }
        printf("%d ",x);
}
```

4. (Aninhamento e Visibilidade - construir.) Implemente em C (após você mesmo pesquisar como) um programa que contenha quatro escopos explícitos de variáveis. Faça os programas mais simples que puder.
Dica: veja como você pode declarar quatro variáveis com o mesmo nome, de tal forma que se tivessem quatro nomes diferentes todas seriam igualmente visíveis em uma mesma parte do programa.

5. Dada a função abaixo:
a) Faça um diagrama mostrando as chamadas;
b) Quantas chamadas serão executadas para avaliar $X(6)$?
c) Indique a seqüência temporal das chamadas.

```
função função X( n ) retorna inteiro
   n :  inteiro;
início
   se n = 0 então
      x ← 0
   senão
      se n = 1 então
         x ← 1
      senão
         se n = 2 então
            x ← 2
         senão
            x ← x(n − 1) + x(n − 2) + x(n − 3)
         fim se
      fim se
   fim se
fim
```

6. (Chamada de função.) Seja a função **raizes_reais()** uma função que retorna o número de raízes reais de um polinômio. Os coeficientes do polinômio estariam armazenados num vetor do tipo **float** que é passado como parâmetro **p**, junto com o grau do polinômio (parâmetro **grau**). Deve também ser passado à função um vetor de **float** para que esta retorne o valor das raízes encontradas.

O protótipo da função é o seguinte:

```
int raizes_reais(float p[], int grau, float r[]);
```

Escreva uma função **main** que, chamando a função **raizes_reais**, imprima na tela todas as raízes reais de um polinômio entrado pelo usuário.

7. (Chamada de função.) Seja a função **linha** a função definida conforme o protótipo abaixo que desenha um segmento de reta na tela gráfica sendo (**x0**,**y0**) as coordenadas do ponto inicial e (**x1**,**y1**) as coordenadas do ponto final do segmento.

```
void linha(int x0, int y0, int x1, int y1);
```

Defina uma função **retangulo** que, utilizando a função **linha**, desenhe um **retangulo** com lados paralelos à borda da tela com centro em um ponto (**xc**,**yc**) e **comprimento** e **altura** passados como parâmetro.

8. (Chamada de função.) Seja a função **troca** cujo protótipo está definido abaixo, uma função que permute o valor de uma variável do tipo **double** por outra, ambas passadas por referência.

Defina uma função **main** que permute os valores das variáveis **a**, **b** e **c**, declaradas abaixo, de forma que no final a<=b<=c. A função **main** deve chamar a função **troca**.

```
void troca(double*x,double*y);
double a, b, c;
```

9. (Chamada de Função) Dada a função *X* definida abaixo, diga qual o valor de *X*(*5; 3*) e quantas chamadas serão feitas nesta avaliação.

```
função X( n, m: inteiro ) retorna inteiro
início
        se (n == m) ou (m == 0) então
                X = 1;
        senão
                X = X(n – 1;m) + X(n - 1;m - 1)
        fim se
fim
```

10. (Recursão – simular) Qual o resultado impresso pelo seguinte programa?

```
#include <stdio.h>
void troca(char s[], int p1, int p2)
{
        char aux;
        aux=s[p1];
        s[p1]=s[p2];
        s[p2]=aux;
}
void pm(char s[],int k, int n)
{
        int i;
        if (k==n) printf("%s,",s);
        else
                for(i=k; i<n; i++)
                {
                        troca(s,k,i);
                        pm(s,k+1,n);
                        troca(s,k,i);
                }
}
main()
{
        char t[5]="abcd";
        pm(t,0,4);
}
```

11. (Recursão – simular) Qual o resultado impresso pelo seguinte programa?

```
#include<stdio.h>
int digitsum(int n)
{
        if (n>0)
             return (n%10) + digitsum(n/10);
        else
        return 0;
}
int digitalroot(int n)
{
        if(n>9) return
                digitalroot(digitsum(n));
        else
                return n;
}
main()
{
        printf("%d %d %d", digitalroot(123), digitalroot(1234), digitalroot(12345));
}
```

12. (Recursão – simular) Qual o resultado impresso pelo seguinte programa?

```
#include <stdio.h>
int g(int n, int a, int b)
{
        if(n==0) return a;
        else return g(n-1,b,a+b);
}
int f(int n)
{
        return g(n,0,1);
}
main()
{
        printf("%d",f(8));
}
```

13. (Recursão – construir.) Defina uma função que calcule de forma recursiva o número de combinações de **n** elementos tomados **m** a **m**, utilizando as seguintes regras:

$$\binom{n+1}{m} = \binom{n}{m} \cdot \frac{n+1}{n-m+1} \qquad \binom{n}{m+1} = \binom{n}{m} \cdot \frac{n-m}{m+1}$$

14. (Recursão – construir.) Defina uma função para fazer busca binária por recursão em um vetor ordenado. A função é definida pelos seguintes passos: (1) Se o vetor tiver tamanho zero retorne -1 para indicar que o elemento procurado não pertence ao vetor nulo; (2) Teste o elemento central do vetor, se for igual ao elemento procurado retorne seu índice; (3) Se o elemento central for diferente do procurado, se for menor procure recursivamente no sub-vetor à esquerda do elemento central e, se for maior, procure-o recursivamente no sub-vetor à direita do elemento central.

15. (Recursão – construir.) Defina uma função que efetue potenciação de forma recursiva.

16. (Recursão Indireta - construir.) Implemente o algoritmo da função que resolve o problema das *Torres de Hanoi* em C. A função deve escrever os movimentos que devem ser feitos com os discos para a solução do jogo. A função deve também retornar a quantidade de movimentos gastos nesta solução.

Dica: Tudo o que é necessário ser feito deve aparecer dentro da função. Não há qualquer variável ou trecho de código que precise ser declarado fora da função solicitada.

17. (Função Iterativa e Recursiva – construir.) Faça um programa que implemente uma função iterativa que calcule em uma vetor de valores inteiros o maior valor, depois faça também outro programa recursivo para o mesmo caso.

18. (Função Iterativa e Recursiva – construir.) Escreva um algoritmo de uma função iterativa para calcular o fatorial de um número. Em seguida, implemente tal função na linguagem C. Faça o mesmo algoritmo utilizando recursão. E, em seguida, implemente tal função na linguagem C.

19. (Função Iterativa e Recursiva – construir.) Faça um algoritmo interativo e um outro recursivo para encontrar o conjunto dos primeiros 10, 20, 50, 100, 200, 500 e 1000 números primos. Use um vetor de inteiros de 32 bits para armazenar o resultado.

20. (Função Iterativa e Recursiva – construir.) Faça um programa que implemente uma função iterativa que calcule o máximo divisor comum e depois faça também outro programa recursivo para o mesmo caso.

21. (Função Recursiva – construir.)Escreva uma versão recursiva do algoritmo de busca binária dado abaixo:

```
declare
    i, j, k :  inteiro;
    a :  vetor[1..n] de T;
início
    i ← 1
    j ← n
    q ← FALSO
    repita
        k ← (i + j) div 2
        se a[k] = x então
            q ← VERDADEIRO
        senão
            se a[k] < x então
                i ← k + 1
            senão
                j ← k − 1
            fim se
        fim se
    até q  ou  (i > j)
fim
```

HELP:

Estagiária: Michelle de Oliveira Parreira

Contato:

Sala 123 IEC

Email: michelle@ita.br