Introdução à Programação de Computadores
GSI002 - 2012/2
Profs. Walter de Oliveira / Renan Cattelan

Prática 9

Funções: passagem de parâmetros por valor/referência e recursividade

Passagem de parâmetros

- Por valor
  - Uma cópia do valor do parâmetro é feita e passada para a função
  - Mesmo que esse valor mude dentro da função, nada acontece com o valor de fora da função
  - Padrão da Linguagem C
- Por referência
  - A referência (seu endereço na memória) da variável é passada à função
  - Qualquer alteração que a variável sofra dentro da função será refletida fora da função
  - Arrays são sempre passados por referência

Passagem por valor

```
01  include <stdio.h>
02  include <stdlib.h>
03
04  void soma _ mais _ um(int n){
05    n = n + 1;
06    printf("Dentro da funcao: x = %d\n",n);
07  }
08
09  int main(){
10    int x = 5;
11    printf("Antes da funcao: x = %d\n",x);
12    soma _ mais _ um(x);
13    printf("Depois da funcao: x = %d\n",x);
14    system("pause");
15    return 0;
16  }
```

```
Saída   Antes da funcao: x = 5
        Dentro da funcao: x = 6
        Depois da funcao: x = 5
```

Passagem por referência

```
01  include <stdio.h>
02  include <stdlib.h>
03
04  void soma _ mais _ um(int *n){
05    *n = *n + 1;
06     printf("Dentro da funcao: x = %d\n",*n);
07  }
08
09  int main(){
10    int x = 5;
11    printf("Antes da funcao: x = %d\n",x);
12    soma _ mais _ um(&x);
13    printf("Depois da funcao: x = %d\n",x);
14    system("pause");
15    return 0;
16  }
```

```
Saída   Antes da funcao: x = 5
        Dentro da funcao: x = 6
        Depois da funcao: x = 6
```

Lista 7

61. Considerando a estrutura

```
struct Vetor{
float x;
float y;
float z;
};
```

para representar um vetor no R3, implemente uma função que calcule a soma de dois vetores. Essa função deve obedecer ao protótipo:

```
void soma (struct Vetor* v1, struct Vetor* v2, struct Vetor* res);
```

onde os parâmetros v1 e v2 são ponteiros para os vetores a serem somados, e o parâmetro res é um ponteiro para uma estrutura vetor onde o resultado da operação deve ser armazenado.

```c
void soma(struct Vetor* v1, struct Vetor* v2, struct Vetor* res) {
    (*res).x = (*v1).x + (*v2).x;
    (*res).y = (*v1).y + (*v2).y;
    (*res).z = (*v1).z + (*v2).z;
}

int main(void)
{
    struct Vetor v1, v2, v3;

    printf("Digite as coordenadas de v1: ");
    scanf("%f %f %f", &v1.x, &v1.y, &v1.z);

    printf("\nDigite as coordenadas de v2: ");
    scanf("%f %f %f", &v2.x, &v2.y, &v2.z);

    soma(&v1, &v2, &v3);

    printf("\nv1 + v2 = \(%.2f, %.2f, %.2f\)\n", v3.x, v3.y, v3.z);

    system("pause");
    return 0;
}
```

Recursão

- Função que chama a si mesma

```
int f(int parametro) {
        ...
        x = f(y);
        ...
}
```

- Critério de parada
  - Determina quando a função deverá parar de chamar a si mesma

Lista 7

14. Faça um algoritmo que receba um número inteiro positivo n e calcule:

    (a) n!

    (b) Somatório de 1 até n

    Obs.: Crie uma função para cada letra.

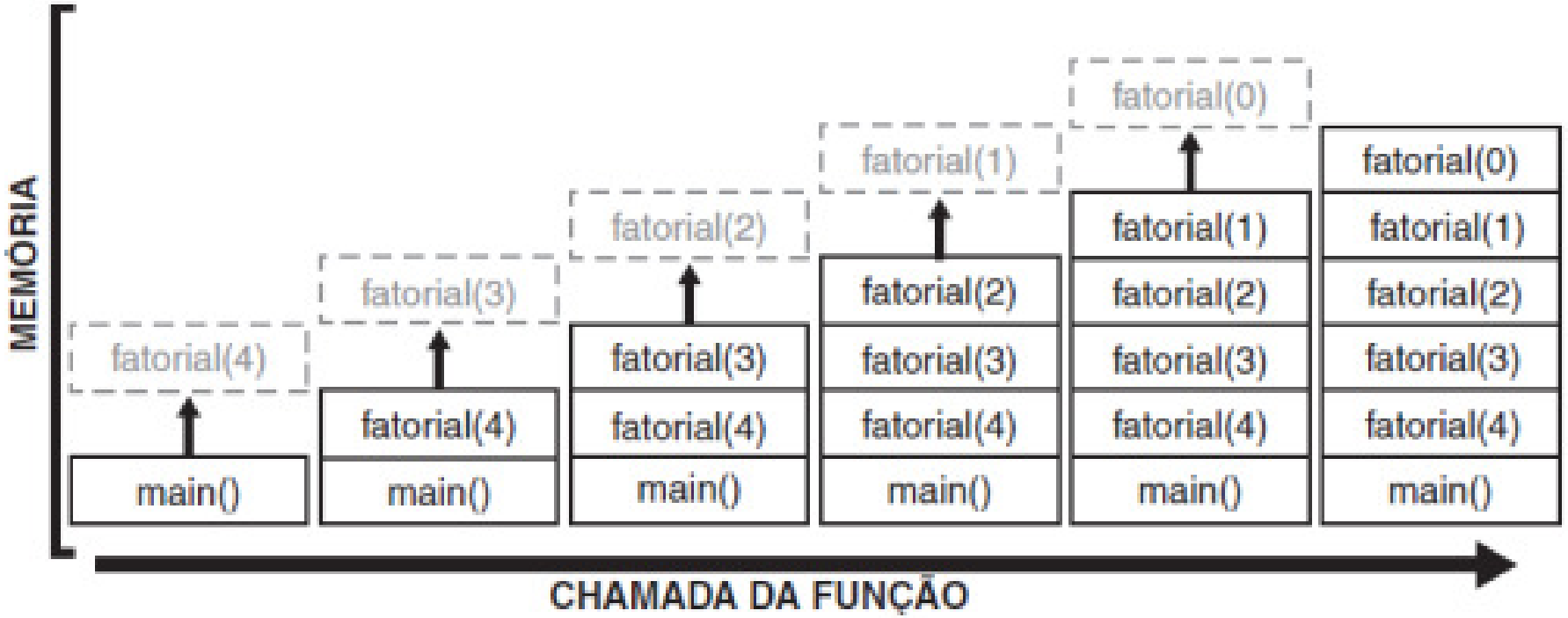
Versão iterativa

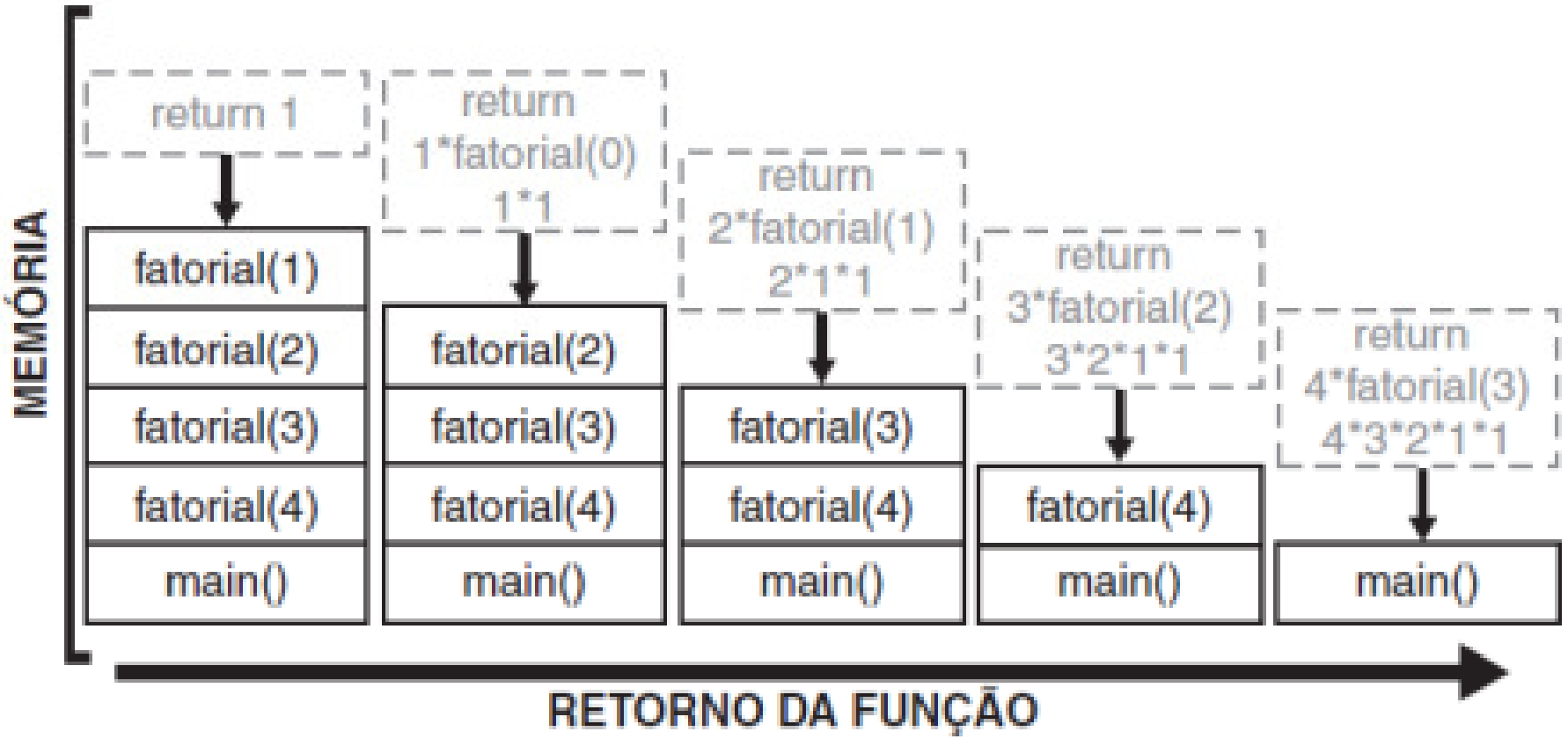
```
int fatorial(int n) {
    if (n < 0) return -1;
    int i, resultado = 1;
    for (i=2; i<=n; i++) {
        resultado = resultado * i;
    }
    return resultado;
}
```

Lista 8

1. Faça uma função recursiva que calcule e retorne o fatorial de um número inteiro N.

```c
#include<stdio.h>

int fatorial(int n) {
    if (n <= 1)
        return 1;
    return n * fatorial(n-1);
}

int main(void)
{
    int n;
    printf("Digite um numero: ");
    scanf("%d", &n);

    printf("\nO fatorial de %d eh: %d\n", n, fatorial(n));

    system("pause");
    return 0;
}
```


MEMÓRIA
CHAMADA DA FUNÇÃO
main()
fatorial(4)
fatorial(3)
fatorial(2)
fatorial(1)
fatorial(0)


MEMÓRIA
return 1
return
1*fatorial(0)
1*1
return
2*fatorial(1)
2*1*1
return
3*fatorial(2)
3*2*1*1
return
4*fatorial(3)
4*3*2*1*1
fatorial(1)
fatorial(2)
fatorial(3)
fatorial(4)
main()
fatorial(2)
fatorial(3)
fatorial(4)
main()
fatorial(3)
fatorial(4)
main()
fatorial(4)
main()
main()
RETORNO DA FUNÇÃO