Recursividade

Prof. Rosana Traversa

Recursividade

- É o nome que se dá quando uma função chama a si própria.
  - Existe a recursão **direta** – quando uma função chama a si mesma diretamente.
  - E a recursão **indireta** – quando uma função chama outra, e esta, por sua vez chama a primeira.

Recursividade

- Uma função pode ser implementada de forma **interativa** ou **recursiva**.
  - quase sempre a forma **recursiva** apresenta uma codificação **mais simples** (reduzida).
  - Por outro lado, implementações **interativas** tendem a ser **mais eficientes** (performance) que as recursivas.

Recursividade

- Sempre que há uma chamada de função (recursiva ou não) os parâmetros e as variáveis locais são empilhadas na pilha de execução.
- No caso da função **recursiva**, para cada chamada é criado um ambiente local próprio. (As variáveis locais de chamadas recursivas são independentes entre si, como se fossem provenientes de funções diferentes).

Recursividade

- Para se aplicar a **recursividade** deve-se pensar na definição recursiva do problema.

- Vejamos o caso do cálculo do fatorial de um número n:
  - Pela definição recursiva temos:

    n! = 1, se n=0 ou

    n*(n-1)!, se n>0

Recursividade - fatorial

- Então teremos a seguinte função:

```
/* calculo do fatorial – função recursiva */
int fatorial (int n) {
    if (n==0) return 1;
    else
        return n*fatorial(n-1);
}
```

Recursividade - Potência

- **Problema**:
  Calcular um valor base (b) elevado a uma potência inteira positiva (p).

- **Definição** em linguagem algoritmica

  se p=0 b^p = 1

  se p>=2 b^p = b*b^(p-1)

Recursividade - Potência

```
/*calculo da potencia – função recursiva */
double potencia (float b, int p) {
    if (p==0) return 1;
    else//desnecessário devido ao return anterior
        return  b*potencia(b, p-1);
}
```

Recursividade - MDC

- Problema:
  Cálculo do mdc entre dois números, usando o algoritmo de Euclides.

quociente		1	1	2
dividendo/divisor	30	18	12	6
resto	12	6	0	

- Lembram-se da versão interativa?

Versão Interativa do MDC

```
int mdc(int a, int b) {
    int r;
    r = a % b;
    while (r ! = 0){
        a = b;
        b = r;
        r = a % b;
   }
 return b;
}
```

Definição do problema do MDC

- Definição recursiva para o MDC:

  mdc(a, b) = b se b divide a,
  ou seja a%b = 0

  mdc(b, a%b) caso contrário

Solução recursiva do MDC

```
int mdc_recursiva(int a, int b) {
        if (a % b == 0) return b;
        return mdc_recursiva (b, a % b);
}
```

Percurso em lista simplesmente ou duplamente encadeada.

- typedef struct lista {
  int info; struct lista *ant, *prox;}
- sendo Lista* p, um ponteiro para o inicio da lista.

  se p==NULL significa: fim da lista, ou lista vazia

  caso contrário escrever conteúdo do item da lista

Solução interativa de percurso em lista

```
void mostra_lista (Lista* p) {
    Lista *aux;
    for(aux=p; aux!=NULL;aux=aux->prox) {
        printf (“%d\t”, aux->info);
 }
```

Solução recursiva de percurso em lista

```
void mostra_lista_recursivo1(Lista* p) {
    if (p==NULL)  return;
    else { //desnecessário após return
        printf (“%d\t”, p->info);
        mostra_lista_recursivo(p->prox);
    }
}
```

OU

Solução recursiva de percurso em lista

```
void mostra_lista_recursivo2 (Lista* p) {
    if (p!=NULL) {
        printf (“%d\t”, p->info);
        mostra_lista_recursivo(p->prox);
    }
}
```

Exercícios:

1. Escreva uma função recursiva que calcule a soma de todos os números compreendidos entre os valores A e B passados por parâmetro.
   Protótipo: **int soma(int a, int b)**
2. Escreva uma função recursiva que calcule os juros compostos de um valor. Para isso o programa deverá ler um valor inicial, o número de meses e a taxa de juros ao mês, e passar estes valores à função como parâmetros.
   Protótipo: **double juroscompostos(double valor, double taxa, int meses)** - formato para double é %lf
3. Escreva uma função que faça a procura sequencial de um valor passado por parâmetro num vetor também passado por parâmetro. Obs. Retornar o índice se encontrado, ou -1 se não encontado.
   Protótipo: **int buscavetor (int *vet, int tam, int valor)**
4. Escreva uma função que faça a procura sequencial de um valor passado por parâmetro numa lista simplesmente encadeada cujo ponteiro para o valor inicial foi passado por parâmetro.
   Protótipo: **void buscalista (Lista *l, int valor)**
5. Escreva uma versão recursiva do bubble sort.
   Protótipo: **void bolha_rec (int *vet, int tam)**