Estruturas de Dados

Profa. Juliana Pinheiro Campos

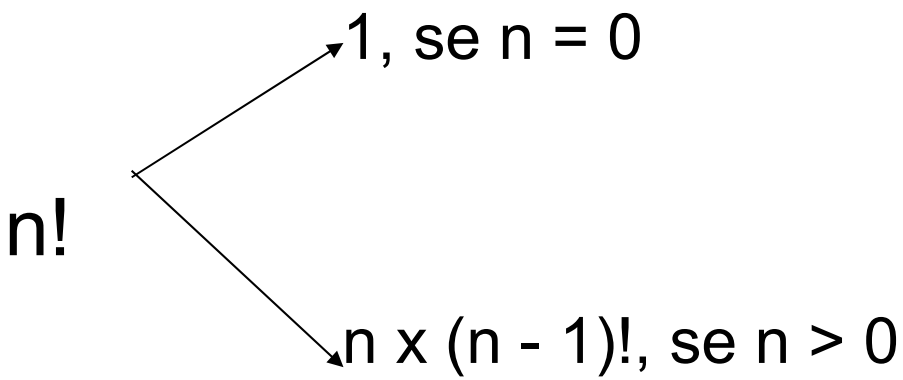
ESTRUTURAS DE DADOS

- Recursividade
  - Funções podem ser chamadas **recursivamente**: Dentro do corpo de uma função podemos chamar novamente a própria função direta ou indiretamente.
  - O uso da recursividade geralmente permite uma descrição mais clara e concisa dos algoritmos, especialmente quando o problema a ser resolvido é recursivo por natureza.

  Diversas implementações ficam mais fáceis com a recursividade. Mas implementações não recursivas tendem a ser mais eficientes.

- Recursividade
  - Cria-se um ambiente local para cada chamada recursiva. As variáveis locais de chamadas recursivas são independentes entre si.
  - As implementações recursivas devem ser pensadas **conforme a definição recursiva do problema.**

- Recursividade
  - Exemplo: Cálculo do fatorial de um número n



- Recursividade
  - Exemplo: Cálculo do fatorial de um número n

```
int fat(int n){
        if(n == 0)
            return 1;
        else
            return n * fat(n – 1);

    }
```

Recursividade

- Uma **exigência fundamental** é que a chamada recursiva de um procedimento P esteja sujeita a uma condição B (**condição de parada ou terminação**), a qual se torna não satisfeita em algum momento da computação.

- Todo procedimento recursivo deve ter uma **condição de parada**, caso contrário entraria em loop infinito.

Recursividade

- Exercício: 7) A série de Fibonacci é definida como: 0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233... onde, os dois primeiros elementos são pré-definidos e do terceiro em diante são gerados à partir da soma dos dois imediatamente anteriores.

  Implemente um programa para calcular o valor do elemento da posicao “n” na série de Fibonacci.
  No programa, implemente uma função recursiva e uma função não recursiva que recebe como parâmetro a posição n do elemento que se deseja gerar. A primeira posição considerada é 0.

ESTRUTURAS DE DADOS

- Pré-processador e macros

  - Antes de ser compilado, o código C passa por um pré-processador que reconhece determinadas diretivas:

  - #include: seguida por um nome de arquivo. O pré-processador a substitui pelo corpo do arquivo especificado. O nome do arquivo incluído é envolto por aspas ou <>.

  - #define: seguida pela definição de uma constante. Toda ocorrência da constante é substituída.

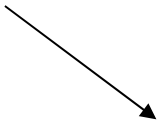
ESTRUTURAS DE DADOS

- Pré-processador e macros
  - Exemplo #define:

```
#define PI 3.14159

float area (float r) {
    float a = PI * r * r;
    return a;

}
```



Substitui toda ocorrência de PI pelo valor definido!

Qual a vantagem de se utilizar a diretiva #define???

Pré-processador e macros

- C permite ainda a utilização da diretiva de definição com parâmetros:

```
#define MAX(a,b) ((a) > (b) ? (a) : (b))
```

- Essas definições com parâmetros recebem o nome de **macros**.

Pré-processador e macros

Cuidado ao utilizar macros:

- Erros de sintaxe podem aparecer na linha em que se utiliza a macro.
- Para evitar efeitos colaterais envolva cada parâmetro da macro com parênteses.

Ex: #define PROD(a,b) (a*b)

```
int main(){
…
printf(“%d”, PROD(3 + 4, 2))
...
}
```