FUNÇÕES

Sistemas de Informação

VAMOS AMADURECER NOSSA VISÃO SOBRE FUNÇÕES ...

Roteiro

01 REVISÃO
Lista

02 PASSAGEM POR REFERÊNCIA

03 RECURSIVIDADE

Passagem de Parâmetros por Referência

- A passagem de parâmetros apresentada até aqui é chamada de **passagem por valor.**
- Nesta modalidade, a chamada da função passa o valor do parâmetro para a função.
- Desta forma, alterações do parâmetro dentro da função não afetarão a variável usada na chamada da função.

PASSAGEM DE PARÂMETROS POR REFERÊNCIA

- No exemplo a seguir a variável f, passada por parâmetro para a função zerar não será alterado dentro da função.

```
#include <stdio.h>

void zerar(float a){
    a = 0;
}

void main(){
   float f;
   f = 20.7;
   zerar(f);
   printf("%d", f);
}
// o valor impresso será 20.7 pois o parâmetro da
//função foi passado por valor.
```

Passagem de Parâmetros por Referência

- Para permitir a alteração da variável usada como parâmetro é preciso passar o endereço da variável, caracterizando desta forma uma passagem por referência.
- Para passar o endereço de uma variável para uma função, deve-se tomar as seguintes providências:
  - na chamada da função deve-se usar o operador & antes do nome da variável;
  - no cabeçalho da função, declarar o parâmetro como um ponteiro;
  - dentro da função, deve-se usar o **operando de derreferência** * para alterar o conteúdo da variável.

Passagem de Parâmetros por Referência

- O exemplo a seguir apresenta uma nova versão da função zerar, desta feita com o uso de um parâmetro passado por referência.

PASSAGEM DE PARÂMETROS POR REFERÊNCIA

```
#include <stdio.h>

void zerar(float *a){ //Define que o parâmetro é uma referência à outra variável
     *a = 0; //o operador de derreferência para alterar o conteúdo da variável
}

void main(){
     float f;
     f = 20.7;
     zerar(&f);  //Passa o endereço da variável f para a função
     printf("%d", f);
// o valor impresso será 0.0 pois o parâmetro da função foi passado por
referência.
          }
```

RECURSIVIDADE

Em uma função recursiva, a cada chamada é criada na memória uma nova ocorrência da função com comandos e variáveis “isolados” das ocorrências anteriores.

Recursividade

- A função é executada até que todas as ocorrências tenham sido resolvidas.
- Porém um problema que surge ao usar a recursividade é como fazê-la parar. Caso o programador não tenha cuidado é fácil cair num loop infinito recursivo o qual pode inclusive esgotar a memória.
- Toda recursividade é composta por um caso base e pelas chamadas recursivas.

RECURSIVIDADE

- **Caso base:** é o caso mais simples. É usada uma condição em que se resolve o problema com facilidade.
- **Chamadas Recursivas:** procuram simplificar o problema de tal forma que convergem para o caso base.

RECURSIVIDADE

Vantagens da recursividade

- Torna a escrita do código mais simples e elegante, tornando-o fácil de entender e de manter.

Desvantagens da recursividade

- Quando o loop recursivo é muito grande é consumida muita memória nas chamadas a diversos níveis de recursão, pois cada chamada recursiva aloca memória para os parâmetros, variáveis locais e de controle.

- Em muitos casos uma solução iterativa gasta menos memória, e torna-se mais eficiente em termos de performance do que usar recursão.

Como decidir se a recursividade é o melhor caminho?

EXEMPLO

```c
//Cálculo de fatorial com função recursiva
#include <stdio.h>
#include <conio.h>

//protótipo da função fatorial
double fatorial(int n);

int main(){
      int numero;
      double f;

      printf("Digite o numero que deseja calcular
      o fatorial: ");
      scanf("%d",&numero);

      //chamada da função fatorial
      f = fatorial(numero);

      printf("Fatorial de %d = %.0lf",numero,f);

      getch();
      return 0;
}
```

```c
//Função recursiva que calcula o
fatorial
//de um numero inteiro n
double fatorial(int n){
      double vfat;
      if ( n <= 1 )
      //Caso base: fatorial de n <= 1
retorna 1
         return (1);
      else{
            //Chamada recursiva
            vfat = n * fatorial(n - 1);
      return (vfat);
 }
}
```

DÚVIDAS?