Recursividade

Para entender o conceito de recursividade, alguns conhecimentos prévios sobre funções devem estar bem compreendidos.

1. **Função**: É um bloco de código que segue uma lógica e realiza determinada tarefa. Esse bloco, também conhecido como subprograma, faz parta da *modularização* de um programa, que consiste em dividir os algoritmos muito grandes em pequenas partes possibilitando reuso e melhor manutenção do código.
   1. **Parâmetros de uma função:** são as variáveis que serão utilizadas pela função para realizar determinada tarefa.
   2. **Chamada de função**: para que uma função seja executada é necessário que a mesma seja chamada dentro do programa principal *[int main()].* A chamada da função é composta pelo nome da função e seus parâmetros.

* Exemplo de função:

Int *soma*(int x, int y){ // função soma, cujo retorno é do tipo inteiro e que recebe os

int sum = x + y; parâmetros X e Y

return sum;

}

1. **Procedimento**: são funções sem retorno, do tipo *void*.
2. **Variável Local:** são variáveis que existem dentro de um subprograma, toda alteração feita nela ocorre localmente e outras funções não tem acesso.
3. **Variável Global:** são variáveis declaradas no inicio do programa, fora de qualquer função. O conteúdo destas variáveis pode ser acessado e alterado por qualquer função do programa. Caso alguma função declare uma variável com o mesmo nome de uma variável global, esta varável se tornará local naquela função, e suas mudanças não ocorrerão na variável principal.
4. **Recursividade**
   1. **O que é?** É uma função que chama a si própria e que age como uma estrutura de repetição.
   2. **Definição formal:** É o mecanismo de programação no qual uma definição de função ou de outro objeto refere-se ao próprio objeto sendo definido.
   3. **Como é composto?** Uma função recursiva necessita de **um** **critério de parada**, para não entrar em um processamento infinito; e **um passo** para dar continuidade no processamento.

\*Exemplo de função recursiva:

int proc(int n){

**if(n>0){** // A condicional (n>0) atua como critério de parada

**return proc(n-1)+proc(n-1);** // O retorno proc(n-1) atua como passo da chamada recursiva

}

return 1;

}

**Referências:**

Tavares, A., Ribeiro, C.; Função e Procedimentos em C - Programação Modular. Disponível em: <http://www2.feg.unesp.br/Home/PaginasPessoais/profandreamarante/disciplinas/aula17_funcoes.pdf> Acessado em: 30/06/2016

Malaquias, J. R.; Programação Funcional – Funções Recursivas. Disponível em: <http://www.decom.ufop.br/romildo/2012-1/bcc222/slides/06-recursividade.pdf> Acessado em: 30/06/2016