- Ponteiro para funções

O C permite que acessemos variáveis e funções através de ponteiros!

Podemos então fazer coisas como, por exemplo, passar uma função como argumento para outra função.

Um ponteiro para uma função tem a seguinte declaração:

*tipo_de_retorno (*nome_do_ponteiro)();*

ou

*tipo_de_retorno (*nome_do_ponteiro) (declaração_de_parâmetros);*

Funções

- Ponteiro para funções

Deve-se ressaltar os parênteses que devem ser colocados obrigatoriamente. Se declaramos:

*tipo_de_retorno *nome(declaração_de_parâmetros);*

Estaríamos, na realidade, declarando uma função que retornaria um ponteiro para o tipo especificado.

Porém, não é obrigatório se declarar os parâmetros da função. Veremos a seguir um exemplo do uso de ponteiros para funções:

```
#include <stdio.h>
#include <string.h>
void PrintString (char *str, int (*func)(const char *));
main (void)
{
  char String []="Curso de C.";
  int (*p)(const char *);
  p=puts;
  PrintString (String, p);
  return 0;
}
void PrintString (char *str, int (*func)(const char *))
{
  (*func)(str);
  func(str);
}
```

Exercício

Construa um programa que receba da linha de comando, com a qual o programa é executado, dois inteiros e a descrição da operação que será efetuada sobre eles (soma, subtracao, multiplicacao, divisao e resto). O programa deve possuir uma função para efetuar cada uma das operações mencionadas, devendo se utilizar do conceito de “ponteiro para função” na seleção da função adequada a ser executada.

- Recursividade

Na linguagem C, assim como em muitas outras linguagens de programação, uma função pode chamar a si mesma. Uma função assim é chamada função recursiva.

Muitos problemas são definidos com base nos mesmos, ou seja, podem ser descritos por instâncias do próprio problema. Para estas classes de problemas, o conceito de recursividade torna-se muito útil.

Funções

- Recursividade (continuação)

Todo cuidado é pouco ao se fazer funções recursivas. A primeira coisa a se providenciar é um critério de parada, o qual vai determinar quando a função deverá parar de chamar a si mesma. Este cuidado impede que a função se chame infinitas vezes.

Um exemplo de um problema passível de definição recursiva é a operação de multiplicação efetuada sobre números naturais. Podemos definir a multiplicação em termos da operação mais simples de adição.

UNIVASF

- Recursividade (continuação)

No caso

A * B

pode ser definido como

A + A * (B - 1)

precisamos agora especificar um critério de parada, qual seria?

A * 0 = 0

Funções

- Recursividade (continuação)

De acordo com o que vimos até o momento, podemos implementar um laço de repetição que implementaria o cálculo da operação de multiplicação com base na operação de adição. Definiremos agora este laço:

```
...
int A, B, RES;
...
for(RES=0;B;B--)
  RES+=A;
...
```