Modularização de Programas

Baseado nos slides de autoria de Rosely Sanches e Simone Senger de Souza

MODULARIZAÇÃO

Um problema complexo é melhor abordado se for dividido primeiramente em vários subproblemas

MODULARIZAÇÃO

A resolução dos subproblemas é feita através de SUBPROGRAMAS

SUBPROGRAMAS

- Os SUBPROGRAMAS devem realizar uma única funcionalidade no programa.
  - possuem “código independente” e devem ser definidos separadamente no programa
    - Na linguagem C a definição do subprograma (função) deve ocorrer antes da função que irá utiliza-lo
- Os SUBPROGRAMAS são declarados no início do programa
  - Prototipação de funções

Estrutura de uma função

```
tipo_da_função nome_da_função(tipo var1, ..., tipo varn)
inicio
   declaração de variáveis locais
   corpo da função
   retorne(variável de retorno)
fim.
```

Prototipação da função

Declaração da função:

```
tipo_da_função nome_da_função(tipo1, ..., tipon)
```

Estrutura de uma função

- Retorna um único valor
- Na definição da função devem ser declarados:
  - o tipo de todos os parâmetros
  - o tipo do valor que a função retorna
  - todas as variáveis utilizadas internamente no subprograma (variáveis locais)
- Para utilizar a função no programa principal basta colocar seu nome (identificador) e os parâmetros reais.

DEFINIÇÃO DE FUNÇÃO

Exemplo

- Dados dois números N e K, calcular a Combinação

$$C_{N,K} = \frac{N!}{K!(N-K)!}$$

- Com a definição de uma função **fat (X)** que calcula o fatorial de um dado X, o cálculo da Combinação fica:

CNK ← **fat (**N**)** / (**fat (**K**)** * **fat (**N-K**)**)

Escopo de variáveis

- Significa a visibilidade de uma variável perante os diversos subprogramas integrantes do programa (ou algoritmo)
  - Variável global: declarada no início do algoritmo ou programa (fora dos subprogramas e programa principal) e é visível por todos.
  - Variável local: declarada dentro de um subprograma e somente visível dentro do mesmo.

```
inteiro troca(inteiro a, inteiro b)
variáveis
  aux: inteiro
inicio
   aux = a
   a = b
   b = aux
fim
Algoritmo principal
variáveis
  a, b: inteiro
inicio
   a = 10; b = 20
   troca(a,b)
   escreva(a,b)
Fim.
```

```
inteiro troca(inteiro a, inteiro b)
variáveis
  aux: inteiro
inicio
   aux = a
   a = b
   b = aux
fim
Algoritmo principal
variáveis
  a, b: inteiro
inicio
   a = 10; b = 20
   troca(a,b)
   escreva(a,b)
Fim.
```

a, b e aux são variáveis locais de troca()

a e b são variáveis locais do alg. princ.

```
inteiro troca(inteiro a, inteiro b)
variáveis
  aux: inteiro
inicio
   aux = a
   a = b
   b = aux
fim
Algoritmo principal
variáveis
  a, b: inteiro
inicio
   a = 10; b = 20
   troca(a,b)
   escreva(a,b)
Fim.
```

a, b e aux são variáveis locais de troca()

Não realiza a troca!

Não é possível retornar dois valores

a e b são variáveis locais do alg. princ.

Escopo de variáveis

- A variável global existe durante toda a execução do programa
  - Interessante quando a mesma é utilizada por várias funções
  - Muitas variáveis globais podem prejudicar a execução do programa (overflow)
  - Os parâmetros enviados para as funções ajudam no entendimento da finalidade da função
    - O uso de variável global deve ser analisado.

Exercício

1. Faça um algoritmo que lê um valor inteiro e positivo n e um algarismo d (0 <= d <=9) e escreve quantas vezes *d* aparece em *n*. Utilize uma função para fazer a contagem.
2. Fazer um algoritmo para calcular a raiz quadrada de um número positivo, usando o roteiro abaixo, baseado no método de aproximações sucessivas de Newton:

   a. Seja y o número:

   i. a primeira aproximação para a raiz quadrada de y é $x_1 = y/2$;

   ii. as sucessivas aproximações serão $x_{n+1} = (x_n^2 + y) / 2x_n$

   b. O algoritmo deverá prever 20 aproximações