WWW.UNICARIOCA.EDU.BR

Algoritmos II

Unidade 1 – Modularização

Profª Giselle Batalha

MELHOR CENTRO UNIVERSITÁRIO DO RIO DE JANEIRO!

Fonte: MEC

Método de Refinamentos sucessivos:

Dividir o problema em vários sub-problemas, evitando assim abordar todos os detalhes do problema simultâneamente.

Nesse contexto, para cada sub-problema será criado uma subrotina, ou seja, um módulo.

Programação Top-Down (ou “de cima para baixo”):

1 - Inicialmente o programador deve saber as tarefas principais do programa, não como fazê-las, apenas quantificá-las;
2 - Depois, modelar como o programa principal irá chamar (gerenciar) essas tarefas;
3 - Então, cada tarefa é detalhada.

UNI CARIOCA Centro Universitário

giselle@unicarioca.edu.br

Vantagens da Programação Top-Down:

- Reutilização de código fonte;
- Facilitar a compreensão de como o problema foi modelado;
- A manutenção é feita por módulo;
- Alterações no módulo valem para todos os lugares onde o módulo é usado;
- O número de linhas de um código fonte de um módulo é relativamente menor, e mais fácil para se entender.

UNI CARIOCA Centro Universitário

giselle@unicarioca.edu.br

MODULARIZAÇÃO

Uma subrotina é um sub-programa com variáveis e comandos próprios e que, para ser executada, precisa ser chamada pela função principal ou outra função.

Em programação temos dois tipos de subrotinas:

- Procedimentos (*procedures*);
- Funções (*functions*).

A função **retorna** um valor, o procedimento **não**.

Na linguagem C temos dois tipos de subrotinas:

- Funções do tipo void, que não retornam valor (procedimentos);
- Funções de um tipo definido, que retornam um valor (funções).

UNI CARIOCA Centro Universitário

giselle@unicarioca.edu.br

EXEMPLO DE PROCEDIMENTO / FUNÇÃO QUE NÃO RETORNA VALOR (VOID)

```
Algoritmo "modularizacao"
procedimento desenha
var
  i:inteiro
inicio
  para i de 0 ate 10 passo 1 faca
       escreva(" # ")
  fimpara
fimprocedimento
Inicio
  desenha
  escreval("  ")
  escreva(" usando funcoes ")
  escreval("  ")
  desenha
Fimalgoritmo
```

```
#include <stdio.h>
 void   desenha( )
{
   int i;
   for (i=0;i<=10;i++)
       printf (" # ");
}
main()
{
    desenha ( );
    printf (" \n usando funções \n");
    desenha ( );
}
```

UNI CARIOCA
Centro Universitário

EXEMPLO DE FUNÇÃO QUE RETORNA VALOR

```
Algoritmo "funcao"
funcao fatorial (n:inteiro):inteiro
var
  i,resultado: inteiro
inicio
   i <- 1
   resultado <- 1
   enquanto i<=n faca
      resultado <- resultado * i
      i <- i+1
   fimenquanto
   retorne resultado
fimfuncao


Inicio
   escreval(" o fatorial de 4 = ",fatorial(4))
   escreval(" o fatorial de 3 = ",fatorial(3))
Fimalgoritmo
```

```
#include <stdio.h>
int fatorial (int  n)
{
   int  i=1, resultado=1;
   while(i<=n)
   {
     resultado*=i;
     i++;
   }
   return  resultado;
}
main()
{
   printf ("\n  o fatorial de 4 = %d", fatorial(4) );
   printf ("\n  o fatorial de 3 = %d", fatorial(3) );
}
```

Tipo: *depende do valor a retornar*

UNI CARIOCA
Centro Universitário

VARIÁVEIS LOCAIS

- Variáveis declaradas dentro de uma função são denominadas locais e somente podem ser usadas dentro do próprio bloco

- São criadas apenas na entrada do bloco e destruídas na saída (automáticas)

VARIÁVEIS LOCAIS

```
void desenha ( )
    {
       int  i, j;
          . . .
    }
main ( )
    {
          int a;
          desenha();
          a = i;  ←  erro
          . . .
    }
```

```
void desenha ( )
{
   int i, j;
     . . .
     . . .
}
void calcula ( )
{
    int i, j;
    . . .
    . . .
}
```

i, j em desenha são variáveis diferentes de i, j em calcula.

UNI CARIOCA
Centro Universitário

Variável que é declarada externamente podendo ser acessada por qualquer função

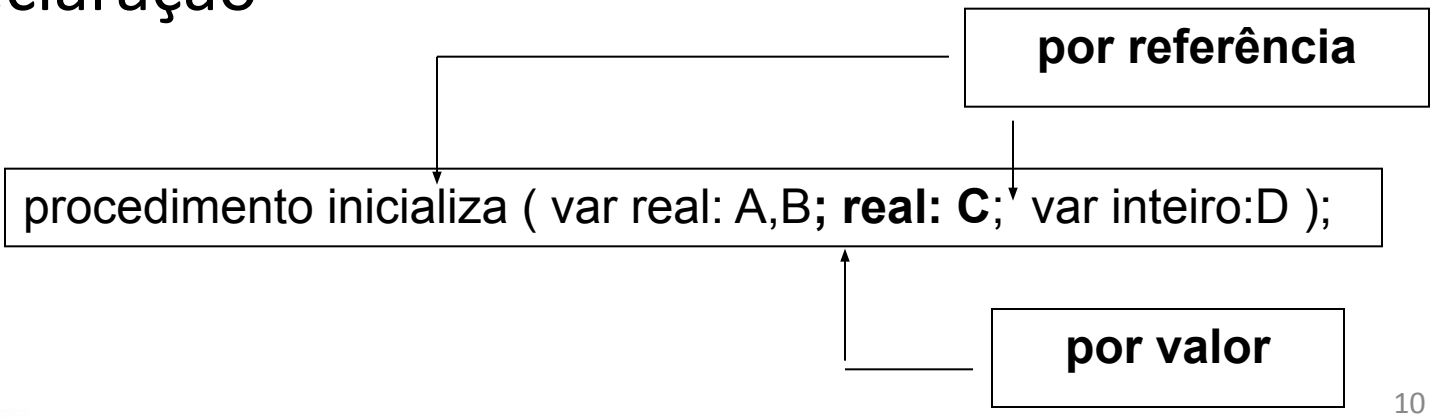
```
#include <stdio.h>
 int i;
main ( )
{
  .........
 .........
desenha ( );
calcula ( );
}
```

```
void desenha ( )
{
     int j;
     i = 0;
     . . .
}
 void calcula ( )
{
      int m;
      i = 5;
      . . .
}
```

Por VALOR: Apenas o valor é transferido. Então, as alterações feitas nos parâmetros formais (da subrotina) não alteram os reais (chamadora).

Por REFERÊNCIA: O endereço do parâmetro real é transferido. Então, as alterações nos parâmetros formais da subrotina na verdade estão sendo feitas sobre os parâmetros reais.

- Declaração



giselle@unicarioca.edu.br

EXEMPLO

```
Algoritmo "Parametros“

procedimento inicializa (var A,B: real; C: real; var D:inteiro)
inicio
  escreval(" Passo 1: ",A,B,C,D)
  A <- 1
  B <- 1
  C <- 1
  D <- 1
  escreval(" Passo 2: ",A,B,C,D)
  fimprocedimento

var
 X,Y,Z: real
 W: inteiro
Inicio
  X <- 0
  Y <- 0
  Z <- 0
  W <- 0
  inicializa(X,Y,Z,W)
  escreval(" Passo 3: ",X,Y,Z,W)

Fimalgoritmo
```

```
#include<stdio.h>
void inicializa (float *A, float *B, float *C, int D)
{
  printf("\n Passo 1: %f %f %f %d",*A,*B,*C,D);
  *A = 1;
  *B = 1;
  *C = 1;
  D = 1;
 printf("\n Passo 2: %f %f %f %d",*A,*B,*C,D);
 }

main()
{
  float X,Y,Z ;
  int W;
  X = 0;
  Y = 0;
  Z = 0;
  W = 0;
  inicializa(&X,&Y,&Z,W);
   printf("\n Passo 3: %f %f %f %d",X,Y,Z,W);
}
```

giselle@unicarioca.edu.br

UNICARIOCA
Centro Universitário

WWW.UNICARIOCA.EDU.BR

1º UNICARIOCA

MELHOR CENTRO UNIVERSITÁRIO DO RIO DE JANEIRO!

Fonte: MEC