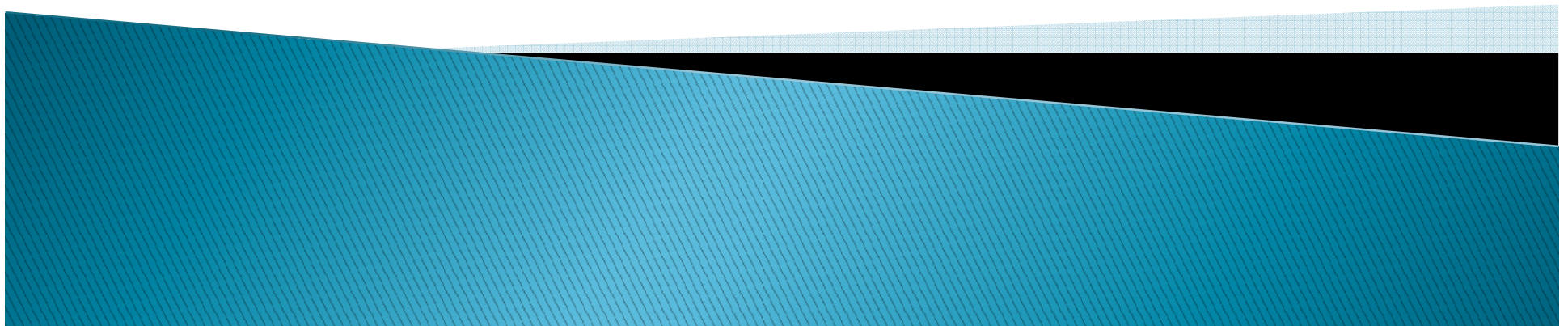


Instituto Federal da Bahia-IFBa
Curso: Engenharia Elétrica
Curso da Linguagem C

Funções e Procedures

Prof. Luiz Cláudio Machado

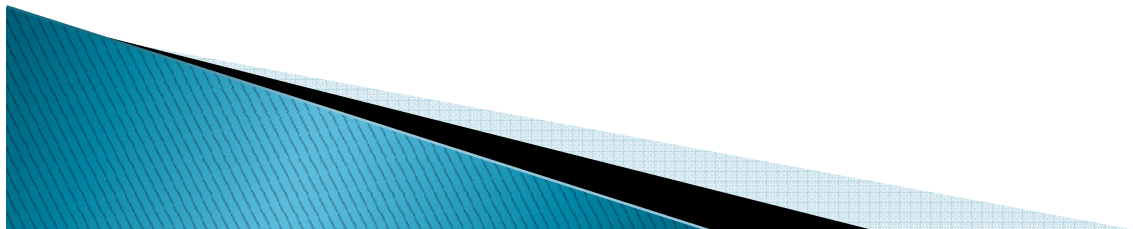


Subprogramas

▶ Subprogramas

Em PASCAL podemos definir blocos de execução internos a um programa, chamados subprogramas. Cada bloco executa uma tarefa determinada. Isto favorece a legibilidade e manutenção do programa, além de auxiliar no domínio da complexidade quando definimos o programa, já que podemos tratar partes do problema de forma isolada. Subprogramas têm parâmetros que podem ser de entrada, saída, ou ambos.

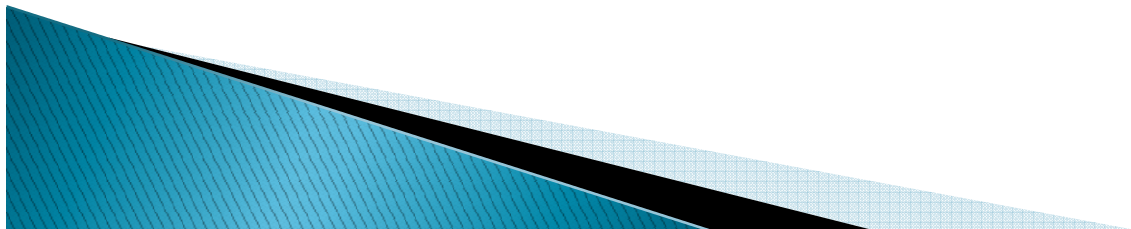
Há dois tipos de subprogramas em Pascal: os procedimentos (PROCEDURES) e as funções (FUNCTIONS).



Diferenças entre Funções e Procedures

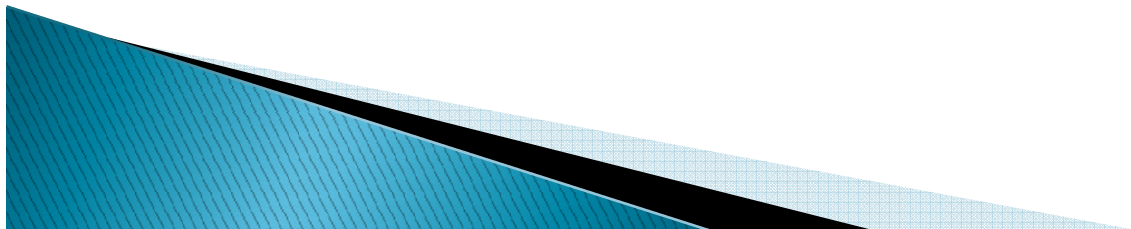
► Diferenças entre Funções e Procedimentos

1. As funções são avaliadas e retornam um valor ao programa que as chama, além dos possíveis parâmetros de saída.
2. Um procedimento não retorna valor nenhum, a função obrigatoriamente retorna um valor a uma determinada variável.
3. Uma função é ativada quando é avaliada uma expressão que a contém, isto é, as funções são utilizadas da mesma forma que as funções predefinidas, como SQR, ORD, LN etc.
4. Um procedimento é ativado através de um comando de chamada do procedimento.



Escopo

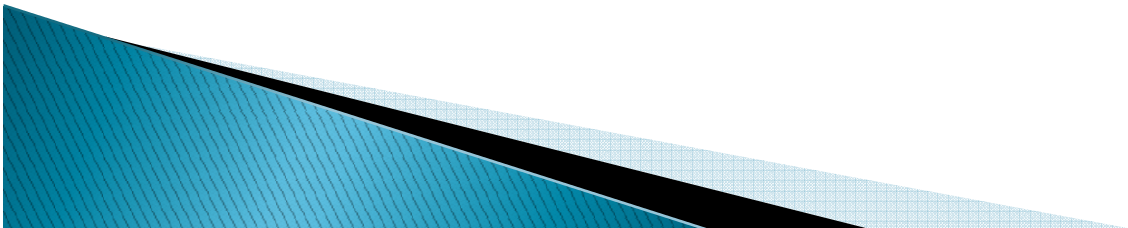
- ▶ EscopoVariáveis Globais: São as variáveis declaradas no programa que são conhecidas em todo programa e inclusive nos subprogramas contidos nele.
- ▶ Variáveis Locais: São as variáveis declaradas em um subprograma, que são conhecidas em todo o subprograma, mas não no programa que contém o subprograma. Caso um subprograma contenha a definição de um outro subprograma (chamemos de "subsubprograma"), as variáveis declaradas no programa e no subprograma serão visíveis no subsubprograma, mas o contrário não é verdadeiro. No escopo do programa não podemos chamar o subsubprograma, assim como não podemos acessar as variáveis do subprograma.
- ▶ Uso de mesmos nomes: se um subprograma definir nomes de variáveis iguais ao do programa principal, ao referenciar uma variável vale a do escopo local.



Exemplo 1

```
/* Corpo principal do programa */  
main ( )  
{  
    clrscr();  
    linha ( ); /* realiza a chamada da função */  
    printf ("\xDB UM PROGRAMA EM C \xDB\n");  
    linha ( );  
    getch();  
}
```

```
/* Declaração da função */  
linha ( )  
{  
    int j;  
    for ( j = 1; j <= 20; j++ )  
        printf ("\xDB");  
    printf ("\n");  
}
```

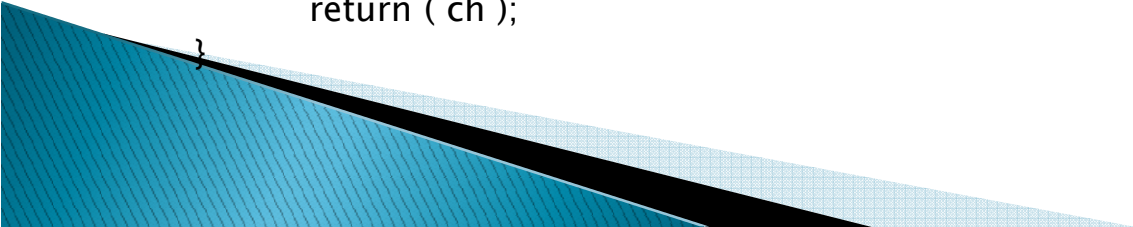


Exemplo 2

```
main ( )
{
    char ch;
    printf ("Digite 'a' e depois 'b': ");
    ch = minusculo ( );
    switch ( ch )
    {
        case 'a' :
            printf ("\n Voce pressionou 'a'."); break;
        case 'b' :
            printf ("\n Voce pressionou 'b'."); break;
        default :
            printf ("\n Voce não obedeceu o que foi solicitado...");
    }
}
```

```
/* Converte para minusculo caso seja digitado uma letra em maiusculo */
minusculo ( )
{
    char ch;
    ch = getche ( );
    if ( ch >= 'A' && ch <= 'Z' )
        ch += 'a' - 'A';

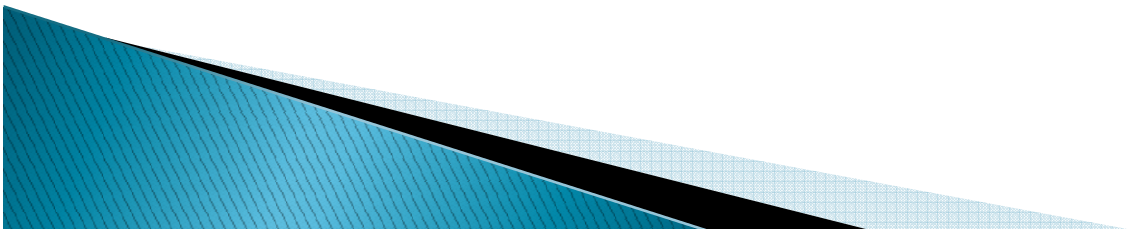
    return ( ch );
}
```



Exemplo 3

```
main ( )
{
    int mins1, mins2;
    printf ("Digite a primeira hora (hora:min): ");
    mins1 = minutos ( );
    printf ("Digite a Segunda hora (hora:min): ");
    mins2 = minutos ( );
    printf ("A diferenca e %d minutos.", mins2 - mins1);
}
```

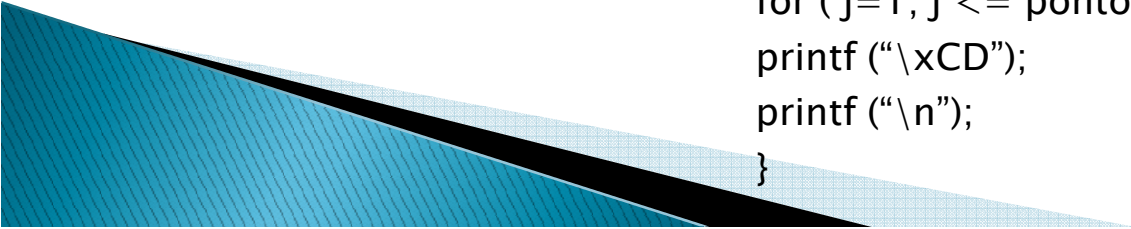
```
/* função minutos */
minutos ( )
{
    int hora, min;
    scanf ("%d:%d", &hora, &min);
    return (hora * 60 + min);
}
```



Exemplo 4

```
void main ( )
{
    printf ("Luiza\t");
    bar(27);
    printf ("Chris\t");
    bar(41);
    printf ("Regina\t");
    bar(34);
    printf ("Cindy\t");
    bar(22);
    printf ("Harold\t");
    bar(15);
}

/* função gráfico de barras horizontal */
bar(pontos)
int pontos; /* variável global da função */
{
    int j; /* variável local da função */
    for ( j=1; j <= pontos; j++ );
    printf ("\xCD");
    printf ("\n");
}
```

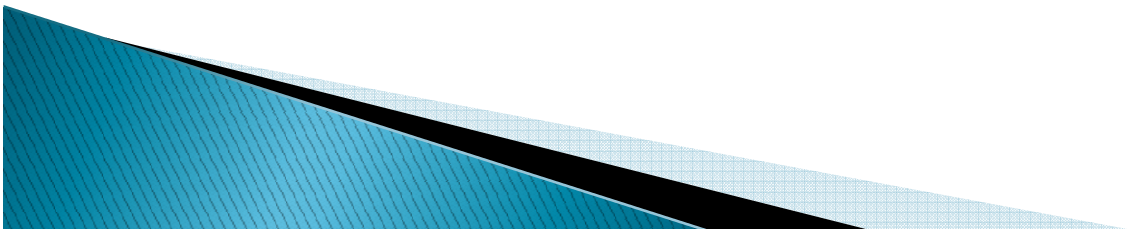


Parâmetros Referência/Valor

Parâmetros No cabeçalho de um subprograma (Funções ou procedimentos) definimos que tipo de dados que serão passados como parâmetros. Há duas formas de passagem de parâmetros de um programa para um subprograma: passagem por *valor* ou passagem por *referência*.

Passagem por valor: o parâmetro formal comporta-se como uma variável local do subprograma, de maneira que as alterações feitas nessa variável dentro do subprograma não tenham efeito sobre o parâmetro real, que pertence ao programa que fez a chamada. Assim, o subprograma só utiliza o valor da variável para um determinado fim, tendo o poder de alterá-la só dentro do subprograma, depois da execução desse subprograma, a variável volta a ter o valor que tinha antes da execução do subprograma.


Passagem por referência: o parâmetro formal comporta-se como se fosse uma variável global, e todas as alterações feitas nesta variável são feitas efetivamente no parâmetro real. Assim, a variável pode ser alterada pelo subprograma e continuar com o valor alterado.



Exemplo por valor

Exemplo 1 – Passagem de parâmetros por valor – Linguagem C

```
01 #include<stdio.h>
02 void troca(int a, int b){
03     int temp;
04     temp=a;
05     a=b;
06     b=temp;
07 }
08 int main(){
09     int a=2,b=3;
10     printf("Antes de chamar a função : \na=%d\nb=%d\n",a,b);
11     troca(a,b);
12     printf("Depois de chamar a função: \na=%d\nb=%d\n",a,b);
13     return 0;
14 }
```



```
torrao@debian: ~/Desktop/wpress
Ficheiro  Editar  Ver  Consola  Separadores  Ajuda
torrao@debian:~/Desktop/wpress$ ./1
Antes de chamar a função :
a=2
b=3
Depois de chamar a função:
a=2
b=3
torrao@debian:~/Desktop/wpress$
```

Exemplo por referência

Exemplo 2 – Passagem de parâmetros por referência – Linguagem C

Contradizendo o título, em C só existe a passagem de parâmetros por valor (obrigatório o uso de apontadores).

```
01 #include<stdio.h>
02 void troca(int *a, int *b){
03     int temp;
04     temp=*a;
05     *a=*b;
06     *b=temp;
07 }
08 int main(){
09     int a=2,b=3;
10     printf("Antes de chamar a função : \na=%d\nb=%d\n",a,b);
11     troca(&a,&b);
12     printf("Depois de chamar a função: \na=%d\nb=%d\n",a,b);
13     return 0;
14 }
```



```
torrao@debian: ~/Desktop/wpress
Ficheiro  Editar  Ver  Consola  Separadores  Ajuda
torrao@debian:~/Desktop/wpress$ ./2
Antes de chamar a função :
a=2
b=3
Depois de chamar a função:
a=3
b=2
torrao@debian:~/Desktop/wpress$
```

Exemplos

```
public class teste
{
    //Este é um metodo semelhante a 'funcao'
    public static int soma(int a, int b)
    {
        int c = a + b;
        return c;
    }

    // Este é um metodo semelhante a 'procedure'
    public static void mudarNome(String primeiroNome)
    {
        primeiroNome += " da silva";
        /*alterei o atributo primeiroNome colocando um sobrenome qualquer
        * o tipo de retorno é 'void', entao o metodo nao retorna nada
        */
        System.out.println(primeiroNome);
    }

    public static void main(String args[])
    {
        int a = 10;
        int b = 20;
        int c = soma(a,b); // o valor de 'c' agora é a soma de 'a' + 'b' = 30
        System.out.println("A soma de a + b = " + c);

        String primeiroNome = "Lula";
        mudarNome(primeiroNome);
    }
}
```