

# Funções

C3

FURG - Fundação Universidade Federal de Rio Grande  
C3 - Centro de Ciências Computacionais

# Motivação

- ▶ Permitir o reaproveitamento de código já construído (por você ou por outros programadores).
- ▶ Permitir a alteração de um trecho de código de uma forma mais rápida. Com o uso de uma função é preciso alterar apenas dentro da função que se deseja.
- ▶ Evitar que os blocos do programa fiquem grandes demais e, por consequência, mais difíceis de entender.
- ▶ Facilitar a leitura do programa-fonte.
- ▶ Separar o programa em partes (subprogramas) que possam ser logicamente compreendidos e testados de forma isolada.
- ▶ Programação Top-down e Bottom-up.

# Procedimentos e Funções

- ▶ São subprogramas que podem ser chamados a partir do programa principal ou de outros subprogramas. Executam uma ou mais tarefas.
- ▶ **Procedimentos** são estruturas que agrupam um conjunto de instruções, que são executadas quando o procedimento é chamado.
- ▶ **Funções** são semelhantes aos procedimentos, exceto que uma função SEMPRE retorna um valor. Um exemplo de função seria o conjunto de instruções para calcular o fatorial de um número e após a função ser executada, ela deve retornar o fatorial do número pedido.



# Notação em C

- ▶ A linguagem C possibilita apenas a definição de funções. A forma geral de uma função em C é:

```
<tipo> NomeDaFuncao ([<lista_de_parâmetros>])  
{  
    /* instrucoes; */  
  
    [return <valor>;]  
}
```

- ▶ Caso o tipo da função não seja fornecido, o C assume que a função é do tipo inteiro.
- ▶ Para simular o comportamento de um procedimento em C, utiliza-se **void** no tipo da função. O *void* é um tipo nulo em C e portanto a função não necessita retornar nenhum valor.

# Exemplo 01 - Procedimento (tipo void)

## Exemplo de procedimento em C:

```
#include <stdio.h>

/* Declaracao - Preciso colocar os ()s mesmo que nao tenha parametros*/
void Ola() {
    /* Corpo */
    printf("\n Ola!!! \n");
}

/* ----- Programa Principal ----- */
int main()
{
    printf("\n Programa Principal !!");

    /* Chamada da funcao */
    Ola();

    printf("\n Posso chamar a funcao de novo:");

    /* Chamada da funcao - Eh necessario chamar com os ()s */
    Ola();
    Ola();

    printf("\n Final do Programa!!");
    return 0;
}
```

## Exemplo 02 - Função de Soma

### Exemplo 2: Função soma em C, sem parâmetros.

```
#include <stdio.h>

/* declaracao da funcao */
int soma1010() {
    /* Corpo da funcao */
    int A;
    A=10+10;

    /* Comando que indica qual o valor que a funcao deve retornar. */
    /* Ao encontrar esse comando, a funcao e' ENCERRADA.          */
    return A;
}

/* ----- Programa Principal ----- */
int main()
{
    int N;
    int Resultado;

    printf("\n Entre com N: ");
    scanf("%d",&N);

    Resultado = N * soma1010() + 20 * soma1010() + 30 * (20+20);

    printf("\n\n  O Resultado e %d ",Resultado);
}
```

## return

- ▶ O comando `return` indica a função qual o valor que ela deve retornar.
- ▶ No exemplo anterior, o `return` indica que o resultado da função deve ser igual ao valor de `A`.
- ▶ Atenção uma função **encerra** assim que encontrar o comando `return`.
- ▶ O comando *return* aceita como argumento constantes, variáveis, expressões, etc. desde que o tipo do argumento seja igual ao tipo da função.



# Parâmetros

- ▶ A fim de tornar mais amplo o uso de funções e procedimentos, permite-se o uso de parâmetros. Estes parâmetros possibilitam que se defina sobre quais dados a função ou o procedimento devem operar.
- ▶ Por exemplo, o programa anterior poderia ser simplificado se a função soma pudesse somar quaisquer dois números, e não apenas 10 mais 10.

# Exemplo 03 - Função com Parâmetros

## Exemplo 3: Função soma com parâmetros.

```
#include <stdio.h>

/* declaracao da funcao */
int soma(int x, int y) {
    /* Corpo da funcao */
    int A;
    A=x+y;

    /* Comando que indica qual o valor que a funcao deve retornar. */
    /* Ao encontrar esse comando, a funcao e' ENCERRADA */
    return A;
}

/* ----- Programa Principal ----- */
int main() {
    int N;
    int Resultado;

    printf("\n Entre com N: ");
    scanf("%d",&N);

    Resultado = N * soma(10,10) + 20 * soma(10,10) + 30 * soma(20,20);

    printf("\n\n   O Resultado e %d ",Resultado);
}
```

# Passagem de Parâmetros - Notação em C

- ▶ A lista de parâmetros em C, é declarada logo após o nome da função e deve estar parênteses. Cada parâmetro deve ser precedido pelo seu tipo e caso haja mais de um, eles devem ser separados por vírgula.
- ▶ Sintaxe:

```
<tipo> nome_funcao (tipo param1, tipo param2, ...) {}
```

- ▶ Exemplos:

```
int soma (int a, int b) {}  
float divide(int a, int b) {}  
float sen(float x) {}  
unsigned long potencia (int x, int y) {}  
char maiuscula(char letra) {}
```

# Função main

- ▶ O programa principal do C é a função `main()` e assim como as outras funções em C, ela deve ter um tipo e no final deve retornar algum valor (`return`).
- ▶ Por isso, em todos os programas que fizemos até agora, era necessário declarar a função `main()` e no seu final encerrá-la com o comando `return`.
- ▶ O tipo padrão da função `main()` é o `int`, mas alguns compiladores aceitam o tipo `void`.

```
/* Declaracao da funcao main() */  
/* Programa Principal do C      */  
int main() {  
    /*...*/  
    return 0; /* Como e uma funcao deve retornar um valor */  
}
```

# Onde Escrever as Funções?

- ▶ Devemos tomar como regra a seguinte afirmativa toda função deve ser declarada antes de ser usada.
- ▶ A declaração de uma função em linguagem C não é exatamente o que fizemos até agora. O que estamos fazendo é a definição da função antes de seu uso. Na definição da função está implícita a declaração.
- ▶ Alguns programadores preferem que o programa principal esteja no início do arquivo fonte. Para isto a linguagem C permite que se declare uma função, antes de defini-la.
- ▶ Esta declaração é feita através do **protótipo** da função.
- ▶ O protótipo da função nada mais é do que o trecho de código que especifica o nome e os parâmetros da função.

## Exemplo 04 - Declaração de Funções

No exemplo a seguir a função SOMA é prototipada antes de ser usada e assim pode ser chamada antes de ser definida.

```
#include <stdio.h>

/* PROTOTIPO da funcao */
/* A definicao da funcao esta depois do main() */
int soma(int x, int y);

/* ----- Programa Principal ----- */
int main() {
    int N;
    int Resultado;

    printf("\n Entre com N: ");
    scanf("%d",&N);

    Resultado = N * soma(10,10) + 20 * soma(10,10) + 30 * soma(20,20);

    printf("\n\n   O Resultado e %d ",Resultado);
}

/* ----- Definicao da funcao ----- */
int soma(int x, int y) {
    int A;
    A=x+y;

    return A;
}
```

# Reutilização de Código

- ▶ Portanto, procedimentos e funções simplificam a codificação e permitem uma melhor estruturação do programa, evitando que uma mesma sequência de comandos seja escrita diversas vezes no corpo da função principal.
- ▶ Por exemplo, suponha um programa para calcular a fórmula  $C(n, p) = \frac{n!}{p!(n-p)!}$  de combinações de  $n$  eventos em conjuntos de  $p$  eventos,  $p \leq n$ .
- ▶ Sem o conceito de funções, teríamos que repetir três vezes as instruções para o cálculo do fatorial de um número  $x$ . Com o conceito de função, precisamos apenas escrever essas instruções uma única vez e substituir  $x$  por  $n$ ,  $p$  e  $(n-p)$  para saber o resultado de cada cálculo fatorial.

## Exemplo 05 - Fatorial e Combinação

```
#include <stdio.h>

/* Prototipo da Funcao */
double fatorial(int x);

/* Definicao da Funcao */
double fatorial(int x) {
    double fat = 1;
    int i;
    for (i=x;i>1;i--)
        fat=fat*i;
    /* Retorna o valor de fat */
    return fat;
}

/* ----- Programa Principal -----*/
int main() {
    int n, p;
    double C;
    scanf("%d %d",&n, &p);
    if ((p>=0)&&(n>=0)&&(p<=n)){

        /* Chamadas da funcao fatorial */
        C= (fatorial(n) / (fatorial(p) * fatorial(n-p)));

        printf("%lf \n", C);
    } else {
        printf("Erro! Valor de n e/ou p invalido!! \n");
    }
    return 0;
}
```



## Exercício 01

Faça uma função que calcula a função:

$f(x) = 5x^4 - 2x^3 + 10x^2 + 5x - 1$ . Onde  $x$  é um valor inteiro.

# Resposta Exercício 01

```
int f(int x) {  
    int fx  
    fx = 5*(x*x*x*x)-2*(x*X*X)+10*(x*x)+5*x-1  
    return fx;  
}
```

## Exercício 02

Faça uma função que recebe por parâmetro o raio de uma esfera e calcula o seu volume ( $v = 4/3 * \pi * R^3$ ).

## Resposta Exercício 02

```
float volume_esfera(int raio) {  
    float volume;  
    volume = ((float) 4/3 * 3.14 * raio * raio * raio);  
    return volume;  
}
```

## Exercício 03

Escreva uma função que recebe as 3 notas de um aluno por parâmetro e uma letra. Se a letra for A o procedimento calcula a média aritmética das notas do aluno e se for P, a sua média ponderada (pesos: 5, 3 e 2).

## Resposta Exercício 03

```
float media (float nota1, float nota2, float nota3, char opcao) {  
    float media=0;  
    if (opcao=='A')  
        media=((nota1+nota2+nota3)/3);  
    if (opcao=='P')  
        media=((nota1*5+nota2*3+nota3*2)/10);  
    return media;  
}
```

## Exercício 04

Faça um procedimento que receba a altura e o peso de uma pessoa, chame uma função para calcular o Índice de Massa Corporal ( $IMC = peso/altura^2$ ) e classifique a pessoa de acordo com a tabela abaixo:

Condição	IMC
Abaixo do Peso	< 18,5
Peso Normal	18,5 <= IMC <= 25
Acima do Peso	25 < IMC <= 30
Obeso	> 30

## Resposta Exercício 04

```
float IMC(int altura, int peso){  
    return (float) (peso/(altura * altura));  
}
```

```
void classifica(int altura, int peso) {  
    float imc = IMC(altura, peso);  
  
    if (imc < 18,5)  
        printf("Abaixo do peso. \n");  
    else  
        if (imc <= 25)  
            printf("Peso normal. \n");  
        else  
            if (imc <= 30)  
                printf("Acima do Peso. \n");  
            else  
                printf("Obeso. \n");  
}
```



# Referências

- ▶ ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. Fundamentos da programação de computadores. Pearson Prentice Hall, 2007.
- ▶ Schildt, Herbert. C completo e total. Pearson Makron Books, 1997.
- ▶ Notas de aula do Prof. Flavio Keidi Miyazawa.
- ▶ Notas de aula do Prof. Emanuel Estrada.
- ▶ Notas de aula do Prof. Alessandro Bicho.