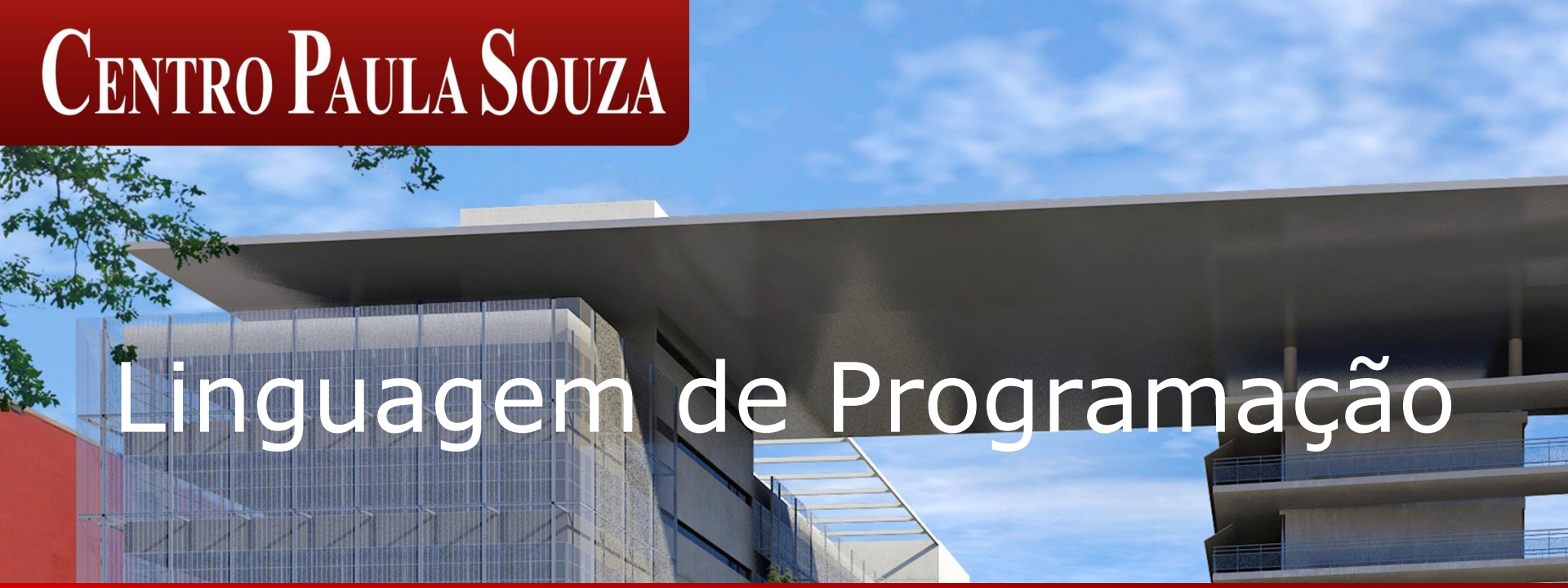


Linguagem de Programação

Estrutura de um programa

- Procedimentos e Funções
- Material: LP_Aula06

Funções em C

Funções

Funções são as estruturas que permitem ao usuário separar seus programas em blocos.

Se não as tivéssemos, os programas teriam que ser curtos e de pequena complexidade.

Para fazermos programas maiores e mais complexos temos de construí-los bloco a bloco.

Sintaxe:

```
  tipo_de_retorno nome_da_função
(declaração_de_parâmetros)
{
      corpo_da_função
}
```

Funções

O **tipo-de-retorno** é o tipo de variável que a função vai retornar.

O default é o tipo **int**, ou seja, uma função para qual não declaramos o tipo de retorno é considerada como retornando um inteiro.

A declaração de parâmetros é uma lista com a seguinte forma geral:

```
tipo nome1, tipo nome2, ... , tipo nomeN
```

Funções

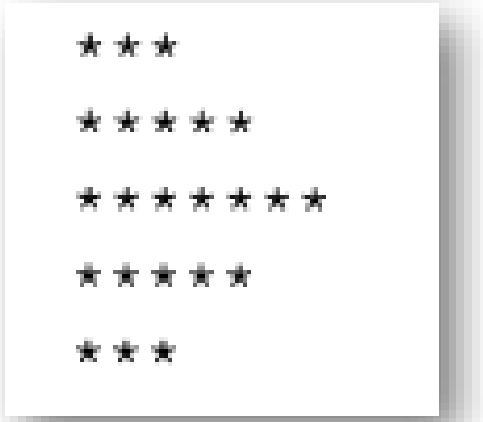
Repare que o tipo deve ser especificado para cada uma das N variáveis de entrada.

É na declaração de parâmetros que informamos ao compilador quais serão as entradas da função (assim como informamos a saída no tipo-de-retorno).

O corpo da função é onde as entradas são processadas, saídas são geradas ou outras operações são feitas.

Considere o seguinte programa (Função – 1)

Escreva um programa que, recorrendo a três funções distintas, escreva na tela a seguinte saída:



```
 1: #include <stdio.h>
 2:
 3: linha3x()
 4: {
 5:    int i;
 6:    for (i=1 ; i<=3 ; i++)
 7:       putchar('*');
 8:    putchar('\n');
 9: }
10:
11: linha5x()
12: {
13:    int i;
14:    for (i=1 ; i<=5 ; i++)
15:       putchar('*');
16:    putchar('\n');
17: }
18:
19: linha7x()
20: {
21:    int i;
22:    for (i=1 ; i<=7 ; i++)
23:       putchar('*');
24:    putchar('\n');
25: }
26:
27: main()
28: {
29:    linha3x();
30:    linha5x();
31:    linha7x();
32:    linha5x();
33:    linha3x();
34: }
```

As funções em negrito são responsáveis individualmente pela escrita dos asteriscos na tela.
A função main() invoca todas as outras funções declaradas anteriormente.

Melhora do programa Anterior (Função – 1)

Note que criamos três funções com tarefas similares:

- linha3x()
- linha5x()
- linha7x()

Poderíamos criar apenas uma função que recebe-se como parâmetro 1 número inteiro, referente ao número de asteriscos que devem ser escritos.

Programa Melhorado

```
 1: #include <stdio.h>
 2:
 3: linha(int num)
 4: {
 5:    int i;
 6:    for (i=1 ; i<=num ; i++)
 7:       putchar('*');
 8:    putchar('\n');
 9: }
10:
11: main()
12: {
13:    linha(3);
14:    linha(5);
15:    linha(7);
16:    linha(5);
17:    linha(3);
18: }
```

Parâmetro: num do tipo inteiro. Significa que a função recebe como argumento um número inteiro passado à mesma.

Parâmetros

A comunicação com uma função se faz através de **argumentos** que são enviados e dos **parâmetros** presentes na função que os recebe.

O número de parâmetros pode ser 0,1,2..., depende das necessidades de programação.

Cada função necessita saber qual o tipo de cada um dos parâmetros.

Exemplo IMC (Função – 2)

Relembrando o IMC.

Criaremos uma função para o cálculo do mesmo.

Exemplo (Função – 2)

```
#include <stdio.h>
#include <stdlib.h>
float imc(float peso, float altura)
{
    float resultado;
    resultado = peso / altura / altura;
    return resultado;
}
int main()
{
    float peso;
    float altura;
    float resultado;
    printf("Calculo do IMC\n");
    printf("Digite o seu peso: \n");
    scanf("%f", &peso);
    printf("Digite a sua altura: \n");
    scanf("%f", &altura);
    resultado = imc(peso, altura);
    printf("Seu imc é igual a: %2.1f \n", resultado);
    system("PAUSE");
    return 0;
}
```

Instrução return

A instrução **return** permite terminar a execução de uma função e voltar ao programa ou função que a invocou.

No exemplo anterior a instrução **return** além de terminar a função, devolve um valor ao programa principal como resultado de sua execução.

Importante: Características de uma função

- Cada função deve ter um nome único.
- Uma função pode ser invocada por outras funções.
- Uma função deve realizar UMA ÚNICA TAREFA bem definida.
- Uma função deve comportar-se como uma caixa preta. Não interessa como funciona, o que importa é o resultado final.
- **O código de uma função deve ser o mais independente possível do resto do programa, e genérico o suficiente para ser reutilizado em outros projetos. (Modularidade).**
- Uma função pode ou não possuir parâmetros, porém deve retornar um valor como resultado do seu trabalho.

Exercício Resolvido

Escreva um programa que solicite dois números ao usuário e apresente na tela como resultado de sua soma e o dobro de cada um deles.

Exercício Resolvido

```
#include <stdio.h>
#include <stdlib.h>
/* Devolve a soma de dois números */
int soma(int a, int b)
{
  return a+b;
}
/* Devolve o dobro de qualquer inteiro */
int dobro(int a)
{
  return a*2;
}

int main(int argc, char *argv[])
{
  int n1, n2, total;
  printf("Digite dois numeros: ");
  scanf("%d%d", &n1, &n2);
  total = soma(n1, n2);
  printf("\nA soma eh %d", total);
  printf("\nO dobro de %d = %d, e de %d = %d\n", n1, dobro(n1), n2, dobro(n2));
  system("PAUSE");
  return 0;
}
```

Exercícios

1) Escreva uma função que receba como parâmetro 2 números inteiros e retorne o maior deles.

2) Escreva uma função que receba como parâmetros 03 números inteiros, referente a três lados de um possível triângulo. Retorne 1 se os valores formarem um triângulo ou 0 se não formarem.

Procedimentos

A diferença entre um procedimento e uma função é que o primeiro não retorna valor ao contrário da função.

Observe o exemplo abaixo:

```
void soma(x,y)
{
    int r;
    r = x + y;
    printf("a soma eh %d", r);
}
```

Exemplo

Para ilustrar a criação de procedimentos, consideremos o cálculo do fatorial de um número.

Podemos escrever uma função que, dado um determinado número inteiro não negativo n, imprime o valor de um fatorial.

Um programa que utiliza este procedimento pode ser escrito conforme próximo slide.

```c
#include <stdio.h>
#include <stdlib.h>
/* programa que imprime um fatorial */
void fat(int n); /* protótipo da função fat() */

/*Função principal ==== */
int main(int argc, char *argv[])
{
  int n;
  printf("Digite um número maior ou igual a zero: ");
  scanf("%d", &n);
  fat(n);
  system("PAUSE");
  return 0;
}

/* Procedure que imprime um fatorial */
void fat(int n)
{
  int i;
  int f =1;
  for(i=1;i<=n;i++)
     f *= i;
  printf("Fatorial do %d eh igual a %d\n", n, f);
}
```

Onde colocar suas funções/procedimentos

Podem ser colocadas em qualquer local do arquivo, antes ou depois de serem invocadas, antes ou depois da função main(). Existe, no entanto, uma restrição que deve ser levada em conta:

- Quando colocar após o main(), deve-se indicar ao compilador qual o cabeçalho (interface ou protótipo) da função, fazendo a declaração da mesma forma que se declaram as variáveis. Não esqueça do “;”.

No exemplo anterior foi feito isso na quarta linha do programa.

```
void fat(int n); /* protótipo da função fat() */
```

Boa prática de programação

Coloque sempre o protótipo das funções no início dos programas, de forma a indicar ao compilador qual a estrutura das funções que vão ser utilizadas pelo programa.

Dessa forma o compilador pode verificar, em cada invocação, se esta (função) foi ou não corretamente implementada.

Exercícios Resolvidos (2)

Escreva a função x_isdigit, que verifica se determinado caractere é dígito ou não. Escreva um programa de teste da função.

Resp. próximo slide.

Resposta

```
 1: #include <stdio.h>
 2:
 3: int x_isdigit(char ch)
 4: {
 5:   return (ch >='0' && ch <= '9');
 6: }
 7:
 8: /* Escreve todos os caracteres não dígitos */
 9:
10: main()
11: {
12:   char c;
13:   while(1) /* Termina com CTRL-C */
14:     {
15:      c=getchar();
16:      if (!x_isdigit(c)) /* se não for digito */
17:         putchar(c);
18:     }
19: }
20:
```

Prof. Me. Marcos Roberto de Moraes, o Maromo

FIM

Referências Bibliográficas

DAMAS, L. M. D. **Linguagem C**. LTC, 2007.

HERBERT, S. **C completo e total**. 3a. ed. Pearson, 1997.