# Programação de Computadores

## Linguagem C

### Funções e Procedimentos

- Funções e procedimentos são blocos de código que podem ser chamados ou invocados a partir de outras partes do programa.
- São usados para modularizar o código, dividindo-o em partes menores e mais gerenciáveis.
- Facilitam a reutilização do código, melhoram a legibilidade e tornam o programa mais organizado.

1. **Funções**:
   - É um bloco de código que recebe argumentos, executa um conjunto de instruções e retorna um valor (ou não retorna valor algum).
   - A declaração de uma função inclui o tipo de dado do valor de retorno, o nome da função e a lista de argumentos que ela espera receber (se houver).

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos

**1.Funções**:

- Aqui está um exemplo de declaração e definição de uma função que retorna a soma de dois inteiros:

```
int somar(int a, int b) {
    int resultado = a + b;
    return resultado;
}
```

- Podem ser chamadas em diferentes partes do programa, passando os argumentos apropriados e armazenando o valor de retorno (se houver) em uma variável.

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos

**1.Procedimentos**:

- É semelhante a uma função, mas não retorna valor algum. Executa um conjunto de instruções e pode receber argumentos.
- Sua declaração é semelhante à de uma função, mas sem o tipo de dado de retorno.
- Aqui está um exemplo de declaração e definição de um procedimento que imprime uma mensagem na tela:

```
void imprimirMensagem() {
    printf("Olá! Esta é uma mensagem de exemplo.\n");
}
```

- Podem ser chamados em diferentes partes do programa, semelhante às funções. São usados para realizar tarefas específicas sem a necessidade de retornar um valor.

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos

- **Declaração e chamada de funções/procedimentos**:
  - Antes de chamar uma função ou procedimento em um programa, é necessário declará-lo ou definir um protótipo.
  - A declaração fornece ao compilador informações sobre o nome da função, o tipo de retorno e a lista de argumentos.
  - Os protótipos de função são geralmente declarados no início do programa, antes de serem chamados.
  - Aqui está um exemplo de declaração de função e procedimento no início do programa:

```
int somar(int a, int b);
void imprimirMensagem();
```

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos

- **Declaração e chamada de funções/procedimentos**:
  - Após a declaração ou definição, as funções e procedimentos podem ser chamados em qualquer lugar do programa, fornecendo os argumentos necessários (no caso de funções) e armazenando o valor de retorno, se houver.
- O uso de funções e procedimentos é fundamental para criar programas mais estruturados, modularizados e fáceis de entender e manter. Eles ajudam a dividir o código em partes menores e mais gerenciáveis, melhorando a legibilidade, a reutilização de código e a manutenção do programa.

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos - Exemplo de código

- Ao executar o programa, ele solicitará ao usuário que digite a largura e a altura do retângulo. Em seguida, a área será calculada pela função e exibida pelo procedimento.
- Esse exemplo ilustra como podemos usar uma função para realizar um cálculo específico (calcular a área do retângulo) e um procedimento para exibir o resultado. Dessa forma, o código fica modularizado e mais organizado.

```c
#include <stdio.h>
// Função para calcular a área de um retângulo
float calcularAreaRetangulo(float largura, float altura) {
    float area = largura * altura;
    return area;
}
// Procedimento para exibir a área de um retângulo
void exibirArea(float area) {
    printf("A área do retângulo é: %.2f\n", area);
}
int main() {
    float largura, altura;
    printf("Digite a largura do retângulo: ");
    scanf("%f", &largura);
    printf("Digite a altura do retângulo: ");
    scanf("%f", &altura);
    // Chamada da função para calcular a área
    float areaRetangulo = calcularAreaRetangulo(largura, altura);
    // Chamada do procedimento para exibir a área
    exibirArea(areaRetangulo);
    return 0;
}
```

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos - Passagem de parâmetros

- **Passagem de parâmetros por valor:**
  - Na passagem de parâmetros por valor, uma cópia do valor da variável é passada para a função ou procedimento. Qualquer modificação feita nos parâmetros dentro da função não afeta a variável original fora da função.
  - A passagem de parâmetros por valor é a forma padrão de passagem de parâmetros em C.
  - Quando uma variável é passada por valor, seu valor é copiado para um novo espaço de memória dentro da função.
  - As alterações feitas nos parâmetros dentro da função são aplicadas apenas a essa cópia, não afetando a variável original fora da função.

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos - Passagem de parâmetros

- **Passagem de parâmetros por valor:**
  - Exemplo:

```c
void dobrarValor(int numero) {
    numero = numero * 2;
}
int main() {
    int valor = 10;
    dobrarValor(valor);
    printf("Valor original: %d\n", valor);  // Resultado: 10
    return 0;
}
```

  - Neste exemplo, a função **dobrarValor( )** recebe um parâmetro numero por valor. A multiplicação por 2 é aplicada apenas à cópia do valor dentro da função. Portanto, o valor da variável valor fora da função permanece inalterado.

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos - Passagem de parâmetros

- **Passagem de parâmetros por referência (ponteiro):**
  - Na passagem de parâmetros por referência, um ponteiro para a variável é passado para a função ou procedimento. Isso permite que a função acesse e modifique diretamente a variável original fora da função.
  - Ao passar um parâmetro por referência, a função recebe o endereço de memória da variável original e pode trabalhar diretamente com ela.
  - As alterações feitas nos parâmetros por referência dentro da função afetam diretamente a variável original fora da função.
  - Para passar um parâmetro por referência, usamos um ponteiro para o tipo de dado correspondente.

# Programação de Computadores

## Linguagem C

### Funções e Procedimentos - Passagem de parâmetros

- **Passagem de parâmetros por referência :**
  - Exemplo:

```c
void dobrarValor(int *pNumero) {
    *pNumero = *pNumero * 2;
}
int main() {
    int valor = 10;
    dobrarValor(&valor);
    printf("Valor: %d\n", valor);  // Resultado: 20
    return 0;
}
```

  - Neste exemplo, a função **dobrarValor( )** recebe um parâmetro ponteiro do tipo **int*** por referência. Ao passar **&valor** como argumento, estamos passando o endereço de memória da variável valor para a função. Dentro da função, utilizamos o operador de desreferência * para acessar e modificar o valor da variável original.
  - Dessa forma, a passagem de parâmetros por referência permite que a função modifique diretamente a variável original fora dela, enquanto na passagem por valor apenas uma cópia do valor é manipulada.

# Programação de Computadores

## Linguagem C

### Escopo de variáveis

- Escopo de uma variável refere-se ao âmbito em que ela pode ser usada, podendo ser variáveis com escopo global ou local.

- As variáveis que são definidas dentro de uma função são denominadas de variáveis com escopo local, ou seja, elas só existem enquanto a função está sendo executada (elas passam a existir quando ocorre a entrada da função e são destruídas ao sair). Uma variável definida dentro de uma função só pode ser usada dentro desta função.

- As variáveis definidas antes da função main() são definidas com escopo global e podem ser utilizadas por todas as funções do código.

# Programação de Computadores

## Linguagem C

### Exemplo

Faça um programa que leia dois valores inteiros, calcule e mostre o resultado da operação de soma. Crie uma função que execute a soma de dois números inteiros e retorne o resultado da operação.

# Programação de Computadores

## Linguagem C

### Referências bibliográficas

MANZANO, J. A. N. G. Programação de Computadores com C/C++. 1. ed. São Paulo: Érica, 2014.

MIZRAHI, Viviane Victorine. Treinamento em Linguagem C. 2. ed. São Paulo: Pearson Prentice Hall, 2008.