Preparatório para Programação

Aula 02 – Apresentação de Funções, Vetores e Ponteiros

Professor Murilo Zanini de Carvalho

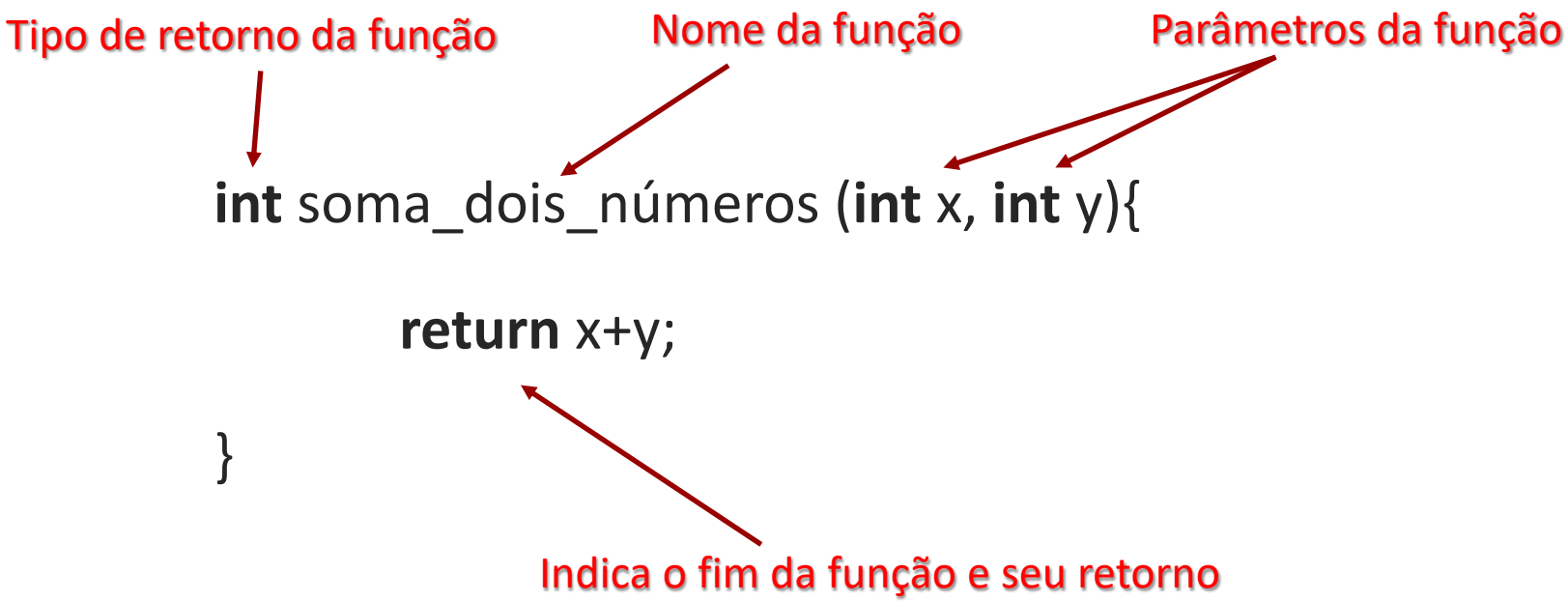
Funções

- São utilizadas para reaproveitar trechos do código em diferentes partes do programa.
- Simplifica a criação de programas e encapsula as lógicas em um único local.
- Facilita a reutilização de códigos.

Premissas de Funções

- As funções devem atender algumas premissas:
  - Não devem realizar mais de uma função.
  - Funções, com exceções das funções específicas para entrada de dados, não devem realizar entrada de dados.
  - Funções, com exceção das funções de exibição, não devem exibir seus resultados, devem retornar eles para que outra parte do programa o faça.

Anatomia de uma função



Tipos de Retorno de uma Função

Tipo	Descrição
int	Retorna um valor inteiro
char	Retorna um caractere
float	Retorna um valor real
void	Não tem nenhum retorno

Função de Soma dois Valores

```
#include <stdio.h>

#include <stdlib.h>

float soma_dois_numeros( float x, float y){

    return x+y;

}
```

Função de Soma dois Valores

```
int main()
{
    float x,y,z,n1,n2,n3;
    printf("Informe dois valores:");
    scanf("%f%f", &x, &y);
    z = soma_dois_numeros(x,y);
    printf("Resultado da soma: %.2f\n", z);
    printf("Informe 2 numeros:");
    scanf("%f%f", &n1, &n2);
    n3 = soma_dois_numeros(n1,n2);
    printf("Resultado da soma: %.2f\n",
n3);
    return 0;
}
```

Função Leitura de Valor Positivo

```
#include <stdio.h>
#include <stdlib.h>

int lerNumeroPositivo(){
   int i;
   do{
      printf("Informe um numero positivo:");
      scanf("%i", &i);
   }while(i<=0);
   return i;
}
```

Função Leitura de Valor Positivo

```
int main()
{
    int a, b, c;
    a = lerNumeroPositivo();
    b = lerNumeroPositivo();
    c = a+b;
    printf("Resultado da soma: %i", c);
    return 0;
}
```

Sistema para ler os valores de entrada de um sensor e apresentar o sinal convertido

```
#include <stdio.h>
#include <stdlib.h>

int leitura_conversor_ad(){
   int sensor;
   do{
      printf("Informe o valor simulado no sensor (0 à 1023):");
      scanf("%i", &sensor);
   }while(sensor < 0 || sensor > 1023);
   return sensor;
}

float converter_sensor(int sensor, int bit, float referencia){
   float escala = referencia / (pow(2,bit)-1);
   return sensor*escala;
}
```

Sistema para ler os valores de entrada de um sensor e apresentar o sinal convertido

```
int main()
{
   int valor;
   float convertido;
   valor = leitura_conversor_ad();
   convertido = converter_sensor(valor, 10, 5);
   printf("Valor convertido: %.2f Volts", convertido);
   return 0;
}
```

Ponteiros

- Recurso mais importante da linguagem, ele permite acessar dados de forma indireta.
- Ponteiros são utilizados para armazenar *ENDEREÇOS*.

Ponteiros

```c
#include <stdio.h>
#include <stdlib.h>

//Programa_Ferias_Ponteiros_01

int main()
{
    int *ptr, x;

    x = 4;

    printf("Valor de X: %i\n", x);

    ptr = &x;   //Atribui o endereço de x para ptr

    *ptr = 5;

    printf("Valor de X: %i\n", x);

    return 0;
}
```

Ponteiros

- Os ponteiros podem ser utilizados para passar valores por referência para funções.
- CUIDADO: quando um parâmetro é enviado por referência, qualquer modificação realizada nele, irá alterar o valor original da variável.

Ponteiros

```c
#include <stdio.h>
#include <stdlib.h>

//Programa_Ferias_Ponteiros_02

void lerValor(int *ptr){
    printf("Informe um valor:");
    scanf("%i", ptr);
}

int main()
{
    int x1, x2, res;
    lerValor(&x1);
    lerValor(&x2);
    res = x1+x2;
    printf("Resultado: %i\n", res);
    return 0;
}
```

Vetores e Matrizes

- Um vetor é um conjunto de variáveis do mesmo tipo que compartilham um mesmo nome;
- Cada uma das variáveis torna-se acessível por meio de seu identificador no vetor.

Elementos do mesmo tipo (Retirado de http://100grana.files.wordpress.com/2009/03/red_power_rangers2.jpg, em 08/09/2011)

Vetores

- Os vetores devem ser declarados para serem utilizados, assim como qualquer outra variável da linguagem C;
- A declaração dos vetores deve acontecer no mesmo local onde é realizada a declaração das variáveis.
- Exemplo:
  - int nome[4];

Vetores

- Para declarar um vetor devemos informar seu tipo, seu nome (mesmo conjunto de regras para dar nomes a variáveis) e, entre colchetes [], seu tamanho;
- Os cochetes são utilizados com os vetores para indicar seu índice;
- É por meio do índice fornecido ao vetor que é possível identificar qual elemento será utilizado.

Exercícios com Funções

- Escreva uma função que receba por parâmetro a altura e o raio de um cilindro circular e retorne o volume desse cilindro. O volume de um cilindro circular é dado por :

$$A = \pi r^2 * h$$

- Elabore um função que receba rês números interios como parâmetros, representando horas, minutos e segundos. A função deve retornar esse horário convertido em segundos.

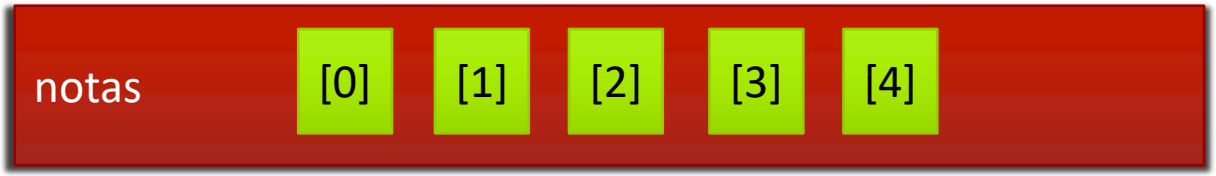
Exemplo 1 - Vetores

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  int vet[3];
  printf(“Informe 2 numeros:”);
  scanf(“%i%i”, &vet[0], &vet[1]);
  vet[2] = vet[0] + vet[1];
  printf(“O resultado eh: %i\n”, vet[2]);
  return 0;
}
```

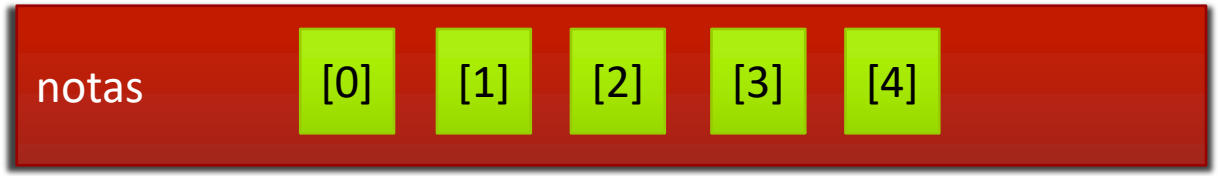
Vetores

- O programa declara um vetor com 5 posições;
  - float notas[5];
- OS VETORES DEVEM SEMPRE POSSUIR UM TAMANHO CONSTANTE.



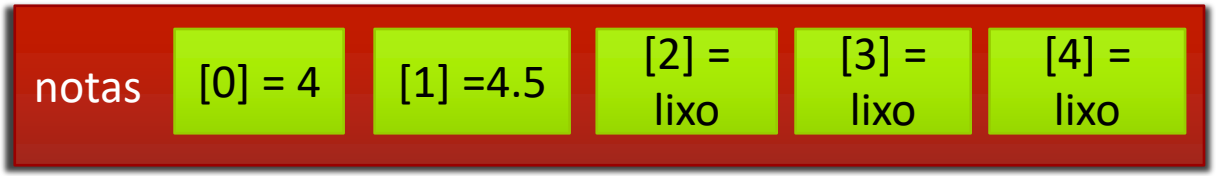
Vetores

- Tomar cuidado: o primeiro indice de um vetor é a posição 0 e a ultima posição é o tamanho do vetor menos 1.



Vetores

- Utilizando um laço de repetição, são atribuídos os valores para cada uma das posições do vetor.
  - notas[0] = 4;
  - notas[1] = 4.5;



Vetores

- VANTAGENS:
  - Redução do tamanho do código;
  - Permite melhor organização do código;
  - Permite realizar expanções do código com maior facilidade que utilizando variáveis independentes;
  - Permite melhor gerenciamento sobre as variáveis utilizadas no código.

Cuidados ao Utilizar Vetores

- CUIDADOS:
  - Ao trabalhar com vetores, deve-se tomar cuidado para não indicarmos um valor fora do range do vetor (seu tamanho);
  - Caso isso venha a acontecer, nenhum erro de compilação será gerado (varia em função do compilador utilizado), mas o funcionamento do programa não será o correto.

Vetores e Funções

- Vetores são passados SEMPRE por referência para funções, portanto, qualquer mudança no vetor dentro da função, vai resultar em uma mudança nele fora dela também.

Tipos Definidos pelo Usuário

- Structs;
- Utilizados para colocar dados diversos dados em uma única representação.
- Sintaxe:

```
typedef struct{
        int a, b, c;
        float area;
} triangulo;
```

Exercícios com Funções

- Elabore uma função que calcule o fatorial de um número inteiro fornecido como parâmetro.

- Faça uma função que receba como parâmetro valor de um ângulo em graus e calcule o vetor do seno desse ângulo usando a sua respectiva série de Taylor:

$$sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

Não se esqueça de converter o valor de x para radianos, utilizando uma outra função auxiliar.

Exercícios com Funções

- Construir um sistema capa de determinar as cores para formar o valor de um resistor.
- Considerações:
  - O sistema deve aceitar valores de resistência e, ao falar as cores necessárias para o resistor, deve perguntar se o usuário deseja ou não continuar.
  - Os resistores são todos de 4 faixas, sendo que apenas valores iguais ou superiores a 1 podem ser informados.
  - Valores inválidos de resistência (que não podem ser calculados) devem ser apresentados com a mensagem “Valor Inválido”.

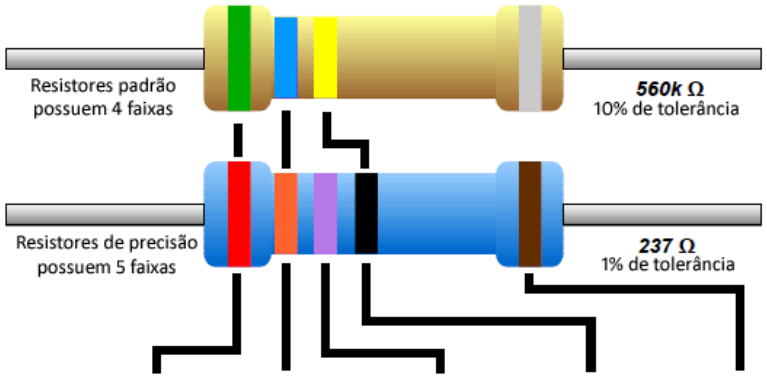
Exercícios com Funções

- Construir seu programa implementando as seguintes funções:
  - int EntradaDeValor() – realiza a leitura do valor da resistência. Em casos de valor inválido de resistência, retorna -1, caso contrário, devolve o valor da resistência em ohms.
  - void MostrarCor(int valor) – exibe a cor da faixa informada.

Tabela com Código de Cores

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

Adaptado de (http://unevcomputacao.com.br/wp-content/uploads/2015/04/codigo_de_cores_resistores.png), em 05.04.2016

Exercícios com Funções

- int primeiraFaixa(int valor) – retorna o valor da primeira faixa.
- int segundaFaixa(int valor) – retorna o valor da segunda faixa.
- int terceiraFaixa(int valor) – retorna o valor da terceira faixa.

Referências

- SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madsen. “Algoritmos”, Ed Makron Books, 1998
- BACKES, André. “Linguagem C: completa e descomplicada”, Ed Elsevier, 2013
- BARRY, Paul; GRIFFITHS, David. “Use a Cabeça! Programação“. Editora Alta Books - 2010

Referências

- BOLTON, W.. “Mecatrônica – Uma abordagem multidisciplinar”, 4 ed., Ed Bookman, Porto Alegre, 2010;
- SANTIAGO, Rafael de. DAZZI, Rudimar Luís. “Ferramenta de apoio ao ensino de algoritmos”, s.d.
- GONDIM, Halley Wesley A.S.. AMBRÓSIO, Ana Paula. “Esboço de Fluxogramas no Ensino de Algoritmos”, s.d.