

Fundação Hermínio Ometto

Bacharelado em Sistemas de Informação

SIF009 - Linguagem de Programação I

Prof. Dr. Sérgio Luis Antonello

Aula 09

29/09/2020

Plano de ensino

1. Unidade I - Programação estruturada e Linguagem C (objetivos b, c e d).
 - 1.1. Conceitos de programação estruturada.
 - 1.2. Estrutura de um programa de computador.
 - 1.3. Códigos fonte, objeto e executável.
 - 1.4. Biblioteca de códigos.
 - 1.5. Compiladores e Interpretadores.
 - 1.6. Processos de compilação e link edição.
 - 1.7. Identificação dos tipos de erros e alertas (léxicos, sintáticos e semânticos).
 - 1.8. Depuração de código.
 - 1.9. Palavras reservadas.
 - 1.10. Tipos de dados.
 - 1.11. Constantes. Variáveis simples e estruturadas. Escopo de variáveis.
 - 1.12. Operadores e precedência.
 - 1.13. Expressões aritméticas, lógicas e relacionais.
 - 1.14. Comandos.
 - 1.15. Ambientes de desenvolvimento e programação.
2. Unidade II - Estruturas de controle (sequência, decisão e repetição), registro e arquivo (objetivos a, c, d).
 - 2.1. Comandos if e switch.
 - 2.2. Comandos for, while e do while.
 - 2.3. Blocos de comandos e aninhamento.
 - 2.4. Definição de tipos.
 - 2.5. Registro.
 - 2.6. Arquivo: leitura e gravação de dados em disco.
3. Unidade III - Ponteiros e Funções (objetivos a, c, d, e).
 - 3.1. Ponteiros.
 - 3.2. Funções.
 - 3.3. Passagem de parâmetro por valor.
 - 3.4. Passagem de parâmetro por referência.
4. Unidade IV - Strings e Variáveis indexadas (objetivos a, c, d).
 - 4.1. Manipulação de strings.
 - 4.2. Manipulação de caracteres.
 - 4.3. Declaração e manipulação de vetores.
 - 4.4. Declaração e manipulação de matrizes.

Plano de ensino

| Data | Atividade |
|-------|-----------|
| 04/08 | Aula 01 |
| 11/08 | Aula 02 |
| 18/08 | Aula 03 |
| 25/08 | Aula 04 |
| 01/09 | Aula 05 |
| 08/09 | Aula 06 |
| 15/09 | Prova 1 |
| 22/09 | Aula 08 |
| 29/09 | Aula 09 |
| 06/10 | Aula 10 |

| Data | Atividade |
|-------|-----------------------------|
| 13/10 | Aula 11 |
| 20/10 | Maratona FHO de Programação |
| 27/10 | Aula 13 |
| 03/11 | Aula 14 |
| 10/11 | Aula 15 |
| 17/11 | Prova 2 Entrega Trabalho |
| 24/11 | Prova SUB |
| 01/12 | Semana Científica |
| 08/12 | Aula 19 |
| 15/12 | Aula 20 |

Sumário da aula

Primeiro momento (revisão)

- ✓ Registro
- ✓ Definição de tipo
- ✓ Leitura e gravação em disco

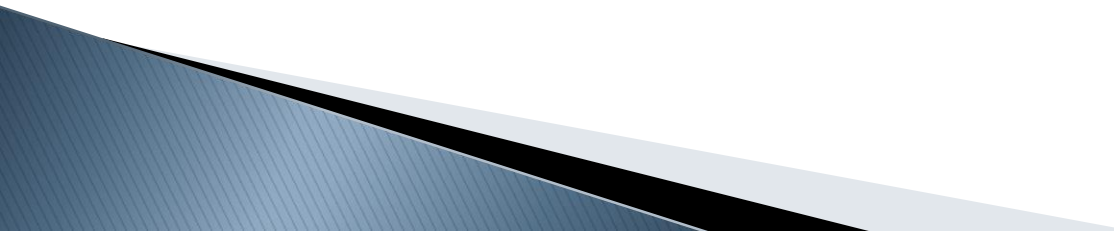
Segundo momento (conteúdo)

- ✓ Funções
- ✓ Escopo de variáveis
- ✓ Retorno de valores
- ✓ Passagem de parâmetro por valor

Terceiro momento (síntese)

- ✓ Retome pontos importantes da aula

1. Primeiro momento: revisão

- ▶ Conceito de registro
 - ▶ Comando que possibilita definir um registro
 - ▶ Comando que possibilita definir um tipo de dados
 - ▶ Arquivo em disco: abrir; ler; gravar; fechar
 - ▶ Correção de exercícios
- 

2. Segundo momento: motivação

- Quantas telas de um sistema bancário usam o CPF?
- Como será uma rotina de validação de CPF?

529.982.247-25

- a) multiplica-se os 9 primeiros dígitos pela sequência decrescente de números de 10 à 2 e soma os resultados;
 - b) Multiplica-se esse resultado por 10;
 - c) 1º dígito verificador = o resto da divisão desse resultado por 11.
 - d) multiplica-se os 10 primeiros dígitos pela sequência decrescente de números de 11 à 2 e soma os resultados;
 - e) Multiplica-se esse resultado por 10;
 - f) 2º dígito verificador = o resto da divisão desse resultado por 11.
- Você já pensou se essa rotina tivesse que ser codificada “em todas as telas”?

2. Segundo momento

- ▶ Funções
- ▶ Escopo de variáveis
- ▶ Retorno de valor
- ▶ Passagem de parâmetro por valor



3. Funções

Conjunto de comandos agrupados em um bloco, que recebe um nome e através deste nome pode ser evocado a qualquer momento.

É um bloco de código que tem uma tarefa específica.

É uma sub-rotina usada em um programa.



3. Funções

Vantagens no uso de funções:

- Permitir o reaproveitamento de código.
- Evitar repetição de um mesmo trecho de código várias vezes.
- Permitir a alteração de um trecho de código de uma forma mais rápida.
- Código mais fácil de entender.
- Dividir um programa em várias partes que realizam tarefas bem definidas.
- Modularização do Código.

3. Funções

Um protótipo de função em linguagem C deve especificar:

- Tipo da Função
- Nome da Função
- Lista de Parâmetros que a Função necessita.

Sintaxe

```
tipo nome_da_funcao (lista_de_parâmetros) {  
    // corpo da função  
    // comandos diversos  
}
```

3. Funções

Localização das funções no código fonte

- Toda função deve ser declarada antes de ser usada.
- A declaração da função deve ocorrer antes do código que a chama para executar. A definição (código completo) pode vir após.

Exemplo de Declaração:

```
void imprimeFrase();
```

Exemplo de Definição:

```
void imprimeFrase(){  
    printf("Hello World");  
}
```

3. Funções: operacionalização

São características da operacionalização do uso de funções:

- O código da função só é executado quando ela é invocada por “alguém”.
- Sempre que uma função é evocada, o programa que a evoca é “suspense” temporariamente.
- Uma vez terminada a execução da função, o controle de execução volta ao local em que ela foi evocada.
- O programa que evoca pode enviar **argumentos** para a função que os recebe no formato de **parâmetros**.

4. Escopo de variáveis

O escopo de uma variável é o bloco de código onde esta variável é válida. Portanto:

- As variáveis valem no bloco que são definidas.
- Variáveis definidas dentro de uma função recebem o nome de variáveis locais (**escopo local**).
- Uma variável definida dentro de uma função **não** é acessível em outras Funções, mesmo que estas variáveis tenham nomes idênticos.
- Variáveis que são declaradas fora das funções tem **escopo global** e dessa forma todas as funções do programa podem acessá-las e manipulá-las.

4. Escopo de variáveis

```
#include <stdio.h>
int varG; ← Global

int imprimeRetorna(){
    int n1;
    n1 = 50;
    varG = 27;
    printf("Valor de n1 dentro da funcao eh %d\n", n1);

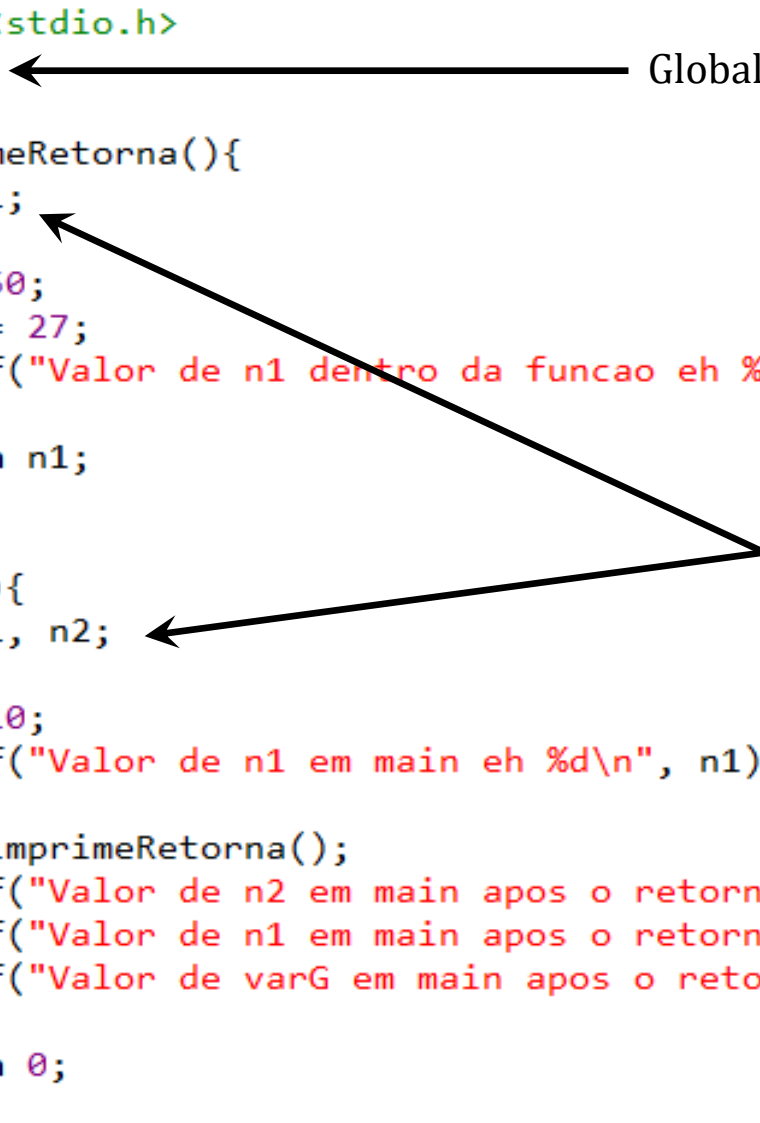
    return n1;
}

int main(){
    int n1, n2; ← Local

    n1 = 10;
    printf("Valor de n1 em main eh %d\n", n1);

    n2 = imprimeRetorna();
    printf("Valor de n2 em main apos o retorno eh %d\n", n2);
    printf("Valor de n1 em main apos o retorno eh %d\n", n1);
    printf("Valor de varG em main apos o retorno eh %d\n", varG);

    return 0;
}
```



5. Retorno de valor

Uma função pode ou não retornar valor para quem a chamou.

- Funções **sem retorno** de valores são declaradas com **void**.
- Funções **com retorno** de valores são declaradas respeitando o tipo de dado que deve ser retornado: **int**, **float**, **etc**.
- Quando fazemos uma chamada a uma função que retorne um valor, devemos **atribuir o valor retornado à uma variável** do mesmo tipo.
- A função retorna um valor para quem a chamou por meio do comando **return**.

6. Passagem de parâmetro

Forma de comunicação entre programas e funções, com envio de dados de um para o outro.

- Durante a declaração de uma função, além do nome, devem-se estabelecer os **parâmetros de comunicação**.
- Variáveis declaradas dentro de uma função apresentam **escopo local**.
- Para um programa usar uma função, basta incluir em seu código uma **chamada** para a função desejada.
- A chamada deve **respeitar a ordem e os tipos** dos parâmetros estabelecidos para a comunicação.

6. Passagem de parâmetro

Forma de comunicação entre programas e funções, com envio de dados de um para o outro.

- A comunicação com uma função se faz através dos **argumentos** que lhes são enviados e dos **parâmetros** presentes na função que os recebe.
- O número de parâmetros depende da necessidade do programador.
- Cada parâmetro tem um tipo específico e deve ser do mesmo tipo do respectivo argumento.

6. Passagem de parâmetro por valor

Sempre que a passagem de parâmetros é realizada por valor, não é a variável ou expressão que é enviada para a função, mas sim uma **cópia do seu valor**.

Se a função alterar o valor recebido (da cópia), o valor da variável original não sofre alteração.



6. Passagem de parâmetro por valor

Exemplo:

```
int Soma (int x, int y) {  
    int res;  
    res = x + y;  
    return res;  
}
```

Parâmetros x e y:
correspondem aos
valores recebidos de
main().

res: valor a retornar para
quem chamou Soma().

```
int main() {  
    int resultado;  
    ...  
    resultado = Soma(3, 6);  
    printf("%d", resultado);  
}
```

Argumentos 3 e 6:
correspondem aos
valores a serem enviados
para Soma().

Valor retornado de
Soma().

6. Passagem de parâmetro por valor

Exemplo

```
int soma(int n1,int n2) {
    int resultado;
    resultado=n1+n2;
    return resultado;
}

int main() {
    int valor1, valor2, valsoma;
    printf("\nEntre com dois números inteiros: ");
    scanf("%d",&valor1);
    scanf("%d",&valor2);
    valsoma = soma(valor1, valor2);
    printf("\n%d + %d = %d\n", valor1, valor2, valsoma);
}
```

6. Passagem de parâmetro por valor

Exemplo

```
float SOMA(float a, float b); // Protótipo da função SOMA

int main() {
    float N1, N2, Result;
    printf("\nEntre com dois números: ");
    scanf("%f", &N1);
    scanf("%f", &N2);
    Result = SOMA(N1, N2);
    printf("\n%.2f + %.2f = %.2f\n", N1, N2, Result);
}

float SOMA(float a, float b) { //Declaração da função SOMA
    float valsoma;
    valsoma = a+b;
    return valsoma;
}
```

7. Exercícios

**Vamos
Programar!**



7. Exercícios

- 1) Escreva uma função que calcule a distância entre dois pontos (x_1, y_1) e (x_2, y_2) digitados pelo usuário em `main()`. Todos os números de entrada e retorno devem ser do tipo `float`.

A função deve receber as coordenadas x, y como parâmetro e retornar a distância entre elas. Entenda estas coordenadas como em um plano cartesiano.

Leia mais em:

<https://mundoeducacao.bol.uol.com.br/matematica/distancia-entre-dois-pontos.htm>

7. Exercícios

2) Altere o exercício anterior, definindo o tipo Ponto como uma Struct.

Escreva uma função que calcule a distância entre dois pontos (x_1, y_1) e (x_2, y_2) digitados pelo usuário em `main()`.

A função deve receber dois pontos como parâmetro e retornar a distância entre eles.



7. Exercícios

- 3) Faça uma função que recebe como parâmetro o raio de uma esfera, calcule o volume na função. Mostre no programa principal o seu volume.

$$V = \frac{4}{3} * \Pi * r^3$$

Leia:

<https://www.todamateria.com.br/a-esfera-na-geometria-espacial/>

<https://mundoeducacao.bol.uol.com.br/matematica/volume-esfera.htm>

- 4) Utilize função com retorno para resolver o problema URI 1103

Alarme despertador.

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1103>

8. Terceiro momento: síntese

As funções são rotinas que resolvem problemas específicos dentro do programa.

A comunicação de dados entre o código principal e uma função ou entre funções ocorre por meio da passagem de parâmetros.

Na passagem de parâmetros por valor, uma cópia do conteúdo do argumento é encaminhado para o parâmetro da função. Se o conteúdo do parâmetro for alterado, o valor do argumento fica preservado.

