

# Lógica de Programação: notas de aula

Prof. Jonatha Costa

2025

# Organização

---

## ① Blocos de programação - C

Funções

Programação em blocos

Conceitos distintos de programação

## ② Exercícios

## ③ Questões propostas

# Objetivo da aula

---

- Estudar a construção de funções para utilização dentro do código principal e utilização de bibliotecas criadas pelo usuário.

# Script

---

Quais elementos de programação estão presentes no *script*?

## Exemplo

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

```
// biblioteca
```

```
int main()
```

```
// Função principal
```

```
{ printf("Oi!");
```

```
// comando da biblioteca
```

```
}
```

- É possível criar bibliotecas e comandos(rotinas) próprias?
- Como criar uma função?
- Como criar uma biblioteca?

# Organização

---

## 1 Blocos de programação - C

Funções

Programação em blocos

Conceitos distintos de programação

## 2 Exercícios

## 3 Questões propostas

# Funções

## Sintaxe

**tipo\_saída** **nome\_da\_funcao** (tipo\_entrada  
var\_entrada)

```
{
<comandos>;
}
```

## Exemplo

```
int soma(int var1, int var2)
{
int res;
res = var1 + var2;
return res;
}
```

No arquivo principal, essa função pode ser definida e declarada em bloco único, ao início de código seguindo a estrutura:

- ① Declaração e definição da função;
- ② Código main.

Ou ainda, essa função pode ser definida ao início de código, antes do main(), seguido-se da definição da função:

- ① Declaração da função;
- ② Código main;
- ③ Definição da função.

# Funções: declaração e definição juntos

---

## Exemplo de função no próprio código

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

```
int soma(int v1,int v2) // Declarando e definindo a função
```

```
{ int res=v1 + v2;  
  return res;  
}
```

```
int main( ) // Programa principal
```

```
{   int res, a, b;  
  printf("Digite um numero a:");  
  scanf("%d", &a);  
  printf("Digite um numero b:");  
  scanf("%d", &b);
```

```
res=soma(a , b); // Chamando a função
```

```
printf("Soma = %d", res);  
}
```

## Funções em modo declaração e definição geminados

Perceba que, neste modo, o *script* contém uma função declarada e já definida no topo de código, e que a função é evocada depois pelo código principal.

### Estrutura:

- 1 Declaração e definição da função;
- 2 Código main.

# Funções: declaração e definição separados

## Exemplo de função no próprio código

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

```
int soma(int v1,int v2) // Declarando a função
```

```
int main( ) // Programa principal
```

```
{ int res, a, b;
```

```
printf("Digite um numero a:");
```

```
scanf("%d", &a);
```

```
printf("Digite um numero b:");
```

```
scanf("%d", &b);
```

```
res=soma(a , b); // Chamando a função
```

```
printf("Soma = %d", res);
```

```
}
```

```
int soma(int v1,int v2) // Definindo a função
```

```
{ int res=v1 + v2;
```

```
return res;
```

```
}
```

## Funções em modo declaração e definição separados

Perceba que, neste outro modo, o *script* contém uma função declarada no topo de código, antes do *main()*, seguindo-se, então a definição da função correspondente à declaração.

### Estrutura:

- 1 Declaração da função;
- 2 Código main;
- 3 Definição da função.



# Estruturas antes ou depois

---

*Note que:*

- **Fluxo do compilador:** Em linguagens como C, o compilador lê o código de cima para baixo. Se o programador utilizar uma função antes de defini-la, o compilador retornará um erro por não saber o que fazer com essa função.
- **Pré-declaração:** A declaração antes do “main” diz ao compilador o que ele precisa saber sobre a função, para que ela possa ser utilizada antes da definição completa.
- **Código mais organizado:** Colocar a declaração da função no início permite que o programador mantenha a função main na parte superior do código, tornando-a mais fácil de encontrar e ler. A declaração também torna o código mais modular, visto que o programador pode definir funções em qualquer lugar, desde que o compilador saiba de sua existência.

# Organização

---

## 1 Blocos de programação - C

Funções

Programação em blocos

Conceitos distintos de programação

## 2 Exercícios

## 3 Questões propostas

# Blocos

---

- As funções podem ser definidas num arquivo biblioteca. Desse modo, o programador pode evocar, no programa **main**, as funções por ele definidas;
- Faz-se necessário, entretanto:

## Configurar os arquivos de blocos

- ❶ Criar o arquivo “.c” (**ScriptDesejado.c**) contendo o *script* desejado;
- ❷ Criar um arquivo de biblioteca com a extensão ‘h’ (**biblioteca.h**) contendo o nome do arquivo “.c”;
- ❸ Incluir no cabeçalho do programa *main* o arquivo **biblioteca.h** e evocar, no *main.c*, o programa (ou função) declarado(a) na biblioteca.

Esse método é usado para particionar um programa grande em blocos menores a fim de melhorar o controle das partes e interações.

# Programa em blocos

“Biblioteca.h”

```

:
void Aula6Ex1();

```

```

:

```

“main.c”

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

```
#include
```

“Biblioteca.h”

```

main()
{
    Aula6Ex1();

```

```

:

```

```

}

```

Script\_desejado.c

```

void Aula6Ex1()
{
    printf("\n* * * * * \n");
    printf("Programando em blocos!");
    printf("\n* * * * * \n");
}

```

## Programa em blocos - Exemplo 2

“calc.h”

```
int soma(int v1, int v2);  
int subtracao (int v1, int v2);  
int divisao (int v1, int v2);  
int multiplicacao (int v1, int v2);
```

“main.c”

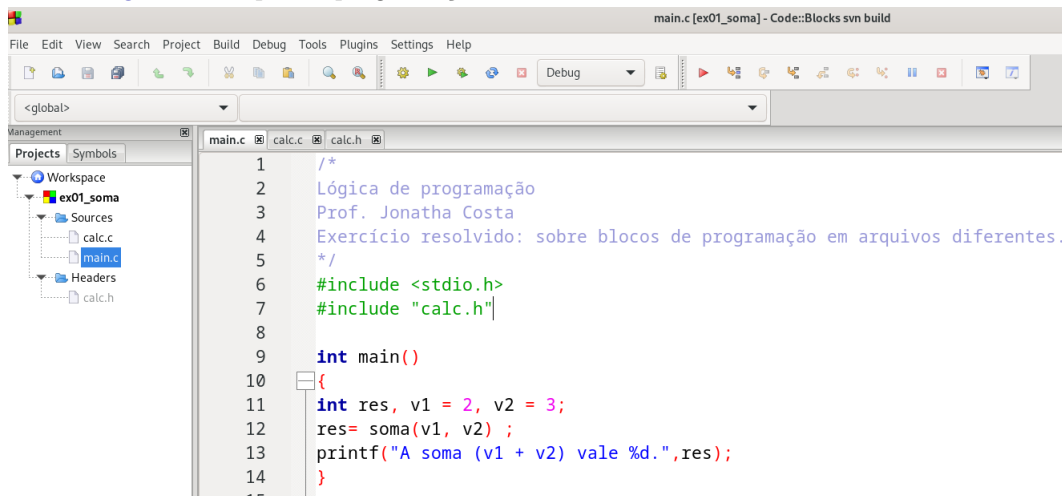
```
#include<stdio.h>  
#include “calc.h”  
int main()  
{   int res;  
  int v1 = 2, v2 = 3;  
  res= soma(v1, v2);  
  printf("A soma (v1 + v2) vale %d.",res);  
}
```

calc.c

```
int soma(int v1,int v2)  
{ return v1+v2;  
}  
  
int subtracao(int v1,int v2)  
{ return v1-v2;  
}  
  
int multiplicacao(int v1,int v2)  
{ return v1*v2;  
}  
  
int divisao(int v1,int v2)  
{ return v1/v2;  
}
```

# Programa em blocos

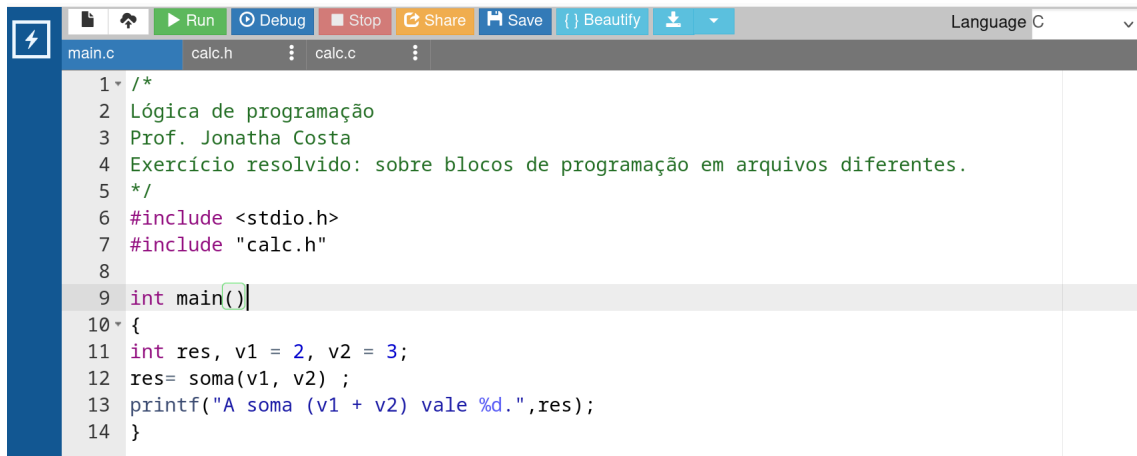
Figura: Exemplo de programação em blocos utilizando o Code Blocks® IDE



Fonte: AUTOR (2024)

# Programa em blocos

Figura: Exemplo de programação em blocos utilizando o onlinegdb<sup>®</sup> IDE



The screenshot shows the onlinegdb IDE interface. The top toolbar includes buttons for Run, Debug, Stop, Share, Save, Beautify, and a Language dropdown set to C. The file explorer shows 'main.c', 'calc.h', and 'calc.c'. The main editor displays the following C code:

```
1 /*
2  Lógica de programação
3  Prof. Jonatha Costa
4  Exercício resolvido: sobre blocos de programação em arquivos diferentes.
5  */
6  #include <stdio.h>
7  #include "calc.h"
8
9  int main()
10 {
11  int res, v1 = 2, v2 = 3;
12  res= soma(v1, v2) ;
13  printf("A soma (v1 + v2) vale %d.",res);
14 }
```

Fonte: AUTOR (2024)

# Organização

---

## 1 Blocos de programação - C

Funções

Programação em blocos

Conceitos distintos de programação

## 2 Exercícios

## 3 Questões propostas



## *Script* com conceitos distintos

---

Um mesmo código pode ser produzido com conceitos diferentes.

- 1 Rotina dentro do programa principal e no mesmo arquivo (*main.c*);
- 2 Rotina fora do programa principal, mas contida no mesmo arquivo (*main.c*);
- 3 Rotina fora do programa principal (*main.c*), mantendo apenas o cabeçalho no arquivo (*main.c*);
- 4 Rotina fora do programa principal (*main.c*); cabeçalho e rotina em arquivos distintos, respectivamente (*rotina.h*) e (*rotina.c*);

Observe com atenção os conceitos e estruturas de cada *script* e apresente as vantagens e desvantagens de cada um.

# Script 01

Figura: Rotina contida no arquivo principal e na função *main()*

```
1  /*
2  Lógica de programação
3  Prof. Jonatha Costa
4  Exercício resolvido: Ler 10 numeros
5  */
6  #include<stdio.h>
7  int main()
8  { int tam_vet=10;
9    int num[tam_vet];
10   for (int i=0;i<tam_vet;i++)
11   {
12     printf("Informe um número (%d / %d): ",i,tam_vet);
13     scanf("%d",&num[i]);
14   }
15
16 }
17
```

Fonte: AUTOR (2024)

## Script 02

Figura: Rotina contida no arquivo principal, porém fora da função *main()*

```
1  /*
2  Lógica de programação
3  Prof. Jonatha Costa
4  Exercício resolvido: Ler 10 numeros
5  */
6  #include<stdio.h>
7
8  void CarregarVetor(int num[],int tam_vet)
9  {
10     for (int i=0;i<tam_vet;i++)
11     {printf("Informe um número (%d / %d): ",i,tam_vet);
12      scanf("%d",&num[i]);}
13 }
14
15 int main()
16 { int tam_vet=10;
17   int num[tam_vet];
18   CarregarVetor(num,tam_vet);
19 }
20
```

Fonte: AUTOR (2024)

## Script 03

(a) Arquivo *main.c*

```
main.c CarregarVetor0.c
1  /*
2  Lógica de programação
3  Prof. Jonatha Costa
4  Exercício resolvido: Ler 10 numeros
5  */
6  #include<stdio.h>
7  // Declaração de cabeçalho
8  void CarregarVetor(int num[],int tam_vet);
9  // Programa principal
10 int main()
11 { int tam_vet=10;
12   int num[tam_vet];
13   CarregarVetor(num,tam_vet);
14 }
15
```

(b) Arquivo *RotCarVetor.C*

```
main.c CarregarVetor0.c
1  // Rotina de carregar vetor
2  #include<stdio.h>
3  void CarregarVetor(int num[],int tam_vet)
4  {
5      for (int i=0;i<tam_vet;i++)
6      {printf("Informe um número (%d / %d): ",i,tam_vet)
7        scanf("%d",&num[i]);
8      }
9  }
10
11
```

Figura: Rotina fora do arquivo principal e da função *main()*

Fonte: AUTOR (2024)

## Script 04

(a) Arquivo *main.c*

```
main.c RotCarVetor.c RotCarVetor.h
1  /*
2  Lógica de programação
3  Prof. Jonatha Costa
4  Exercício resolvido: Ler 10 numeros
5  */
6  #include<stdio.h>
7  #include "RotCarVetor.h"
8
9  int main()
10 { int tam_vet=10;
11   int num[tam_vet];
12   CarregarVetor(num,tam_vet);
13 }
14
```

(b) Arquivo *RotCarVetor.C*

```
main.c RotCarVetor.c RotCarVetor.h
1  #include<stdio.h>
2  void CarregarVetor(int num[],int tam_vet)
3  {
4      for (int i=0;i<tam_vet;i++)
5      {printf("Informe um número (%d / %d): ",i,tam_vet);
6        scanf("%d",&num[i]);
7      }
8  }
9
```

(c) Arquivo *RotCarVetor.h*

```
main.c RotCarVetor.c RotCarVetor.h
1  // Declaração de cabeçalho
2  void CarregarVetor(int num[],int tam_vet);
3
```

Figura: Rotina fora do arquivo principal e da função *main()*. Cabeçalhos no arquivo *RotVet.h*

Fonte: AUTOR (2024)

# Considerações finais

---

Apresente as vantagens e desvantagens de cada conceito de programação contida nos *scripts* acima.

# Organização

---

- 1 Blocos de programação - C
- 2 Exercícios
- 3 Questões propostas

## Exercícios de Estruturas de controle de fluxo

---

**Faça uso de duas ou três estruturas de controle de fluxo para cada item proposto.**

Bloco 01 - Escreva um programa na linguagem C para:

- 1 Ler um número e informar se o número é maior, menor ou igual a 7, 0;
- 2 Ler um número e informar se o número par ou ímpar;
- 3 Ler um número e informar se o número é primo ou não;
- 4 Ler um número e informar se o número pertence aos N;

Bloco 02 - Escreva um programa na linguagem C para:

- 1 Ler 5 valores, encontrar o maior, o menor e a média utilizando números reais (float).
- 2 Ler uma letra e verificar se é uma vogal ou não.
- 3 Leia um número entre 0 e 10, e escreva este número por extenso.
- 4 Elabore um código que receba dois números, a e b tal que  $0 \leq a \leq 10$  e  $25 \leq b \leq 100$ , identifique e informe os valores ímpares de primos contidos nesse intervalo.



## Exercícios de Estruturas de controle de fluxo

---

**Faça uso de duas ou três estruturas de controle de fluxo para cada item proposto.**

Bloco 03 - Escreva um programa na linguagem C para calcular e informar:

$$\textcircled{1} \quad z = \sum_{i=1}^{10} x_i$$

$$\textcircled{2} \quad z = \sum_{i=1}^{10} x_i y_i$$

$$\textcircled{3} \quad z = \sum_{i=1}^{10} (\sqrt{x_i^2 + y_i^2})$$

Onde  $x=\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$  e  $y=\{10, 9, 8, 7, 6, 5, 4, 3, 2, 1\}$ .

## Exercícios de Estruturas de controle de fluxo

---

**Faça uso de duas ou três estruturas de controle de fluxo para cada item proposto.**

Bloco 04 - Escreva um programa na linguagem C para:

- 1 Ler uma matriz de 3 por 3, exibí-la e verificar se esta é triangular inferior e informar ao usuário.
- 2 Ler e preencher uma matriz de 3 por 3, exibí-la e verificar se esta é triangular inferior, superior ou matriz diagonal e informar ao usuário.
- 3 Escrever um programa que retorne ao usuário o k-ésimo dígito da parte não inteira de  $\pi$  e o valor de  $\pi$  até o dígito k-ésimo dígito. Assuma que  $\pi$  tem apenas 13 dígitos em sua parte não-inteira que o usuário desconhece isto.

# Organização

---

- 1 Blocos de programação - C
- 2 Exercícios
- 3 Questões propostas**

## Questões propostas aplicadas à engenharia

---

- **Controle de Temperatura de um Forno**

Um forno industrial precisa manter a temperatura dentro de uma faixa de  $5^{\circ}\text{C}$  em relação à temperatura desejada. Escreva um programa em C que receba a temperatura desejada e a temperatura atual do forno. O programa deve acionar um alarme se a temperatura atual estiver fora da faixa permitida.

- **Monitoramento de Nível de Líquido**

Um tanque de líquidos possui sensores que medem o nível atual de um líquido em mililitros. Escreva um programa em C que monitore o nível do tanque e ative uma bomba de escoamento quando o nível do líquido exceder um determinado limite, e desative a bomba quando o nível estiver abaixo do limite.

- **Aquisição de Dados de um Sensor de Pressão**

Você está implementando um sistema de aquisição de dados para monitorar a pressão em um tubo. Escreva um programa em C que leia os valores de um sensor de pressão a cada segundo e calcule a média desses valores a cada minuto.

## Questões propostas aplicadas à engenharia

---

- **Sistema de Alarme de Incêndio**

Um sistema de alarme de incêndio em um prédio monitora a temperatura e a concentração de fumaça. Escreva um programa em C que ative o alarme se a temperatura ultrapassar 70°C ou se a concentração de fumaça ultrapassar um limite seguro.

- **Controle de Nível de Água em uma Caldeira**

Um sistema de controle precisa manter o nível de água em uma caldeira entre dois valores limites. Escreva um programa em C que monitore o nível de água e ative a entrada de água se o nível estiver abaixo do mínimo e a desligue se o nível estiver acima do máximo.

- **Controle de Iluminação Automática**

Em um sistema de iluminação inteligente, a intensidade das luzes deve ser ajustada automaticamente com base na luz ambiente medida por um sensor LDR (Light Dependent Resistor). Escreva um programa em C que ajuste a intensidade da iluminação interna com base na leitura do sensor LDR.

## Questões propostas aplicadas à engenharia

---

- **Detecção de Obstáculos em um Veículo Autônomo**

Um veículo autônomo utiliza sensores de proximidade para evitar colisões. Escreva um programa em C que analise os dados de múltiplos sensores de proximidade e acione uma mudança de direção ou freio se algum obstáculo for detectado a menos de 1 metro do veículo.

# Exercícios

---

- Veja material auxiliar e lista de códigos em:  
<https://github.com/jonathacosta/PL>