

# **CT62A**

# **COMPUTAÇÃO 1**

## **Aula 04 - Introdução à Linguagem C**

**Profs. Rafael Mantovani e Adalberto Lazarini**



Apucarana - PR, Brasil

Universidade Tecnológica Federal do Paraná (UTFPR)

# Roteiro

- 1 Introdução**
- 2 Estrutura Básica em C**
- 3 Entrada e Saída**
- 4 Referências**

# Roteiro

**1** Introdução

**2** Estrutura Básica em C

**3** Entrada e Saída

**4** Referências

# Introdução

- **Linguagem de programação** é conjunto de ferramentas, regras de sintaxe e símbolos, ou códigos que nos permitem escrever programas de computador
- A primeira e mais primitiva linguagem de computador é a própria linguagem de máquina (0s e 1s)
- Antigamente, um programa era difícil de ser criado, o processo era longo e caro
- Essa complexidade levou à necessidade de desenvolver novas técnicas e ferramentas

# Introdução

- A linguagem C nasceu na década de 70 (Dennis Ritchie) sendo padronizada pela ANSI (1989)
- A linguagem C foi desenvolvida a partir das linguagem B (Ken Thompson) e BCPL (Martin Richards)
  - Inseriu conceito de tipos de dados
  - Independente de hardware
- A linguagem C é genérica e utilizada para a criação de diversos tipos de programas:
  - Planilhas e editores de texto, sistemas operacionais, programas de automação, banco de dados, automação industrial, etc ...

# Introdução

- Comumente chamada de linguagem de “nível médio”
  - Combina elementos de alto nível com funcionalidade assembly (linguagem de baixo nível)
  - Manipula bits, bytes e endereços
- Linguagem portável
  - Possível adaptar o mesmo software a diferentes tipos de computadores
  - **Obs:** desde que siga os padrões (ANSI, etc)

# Roteiro

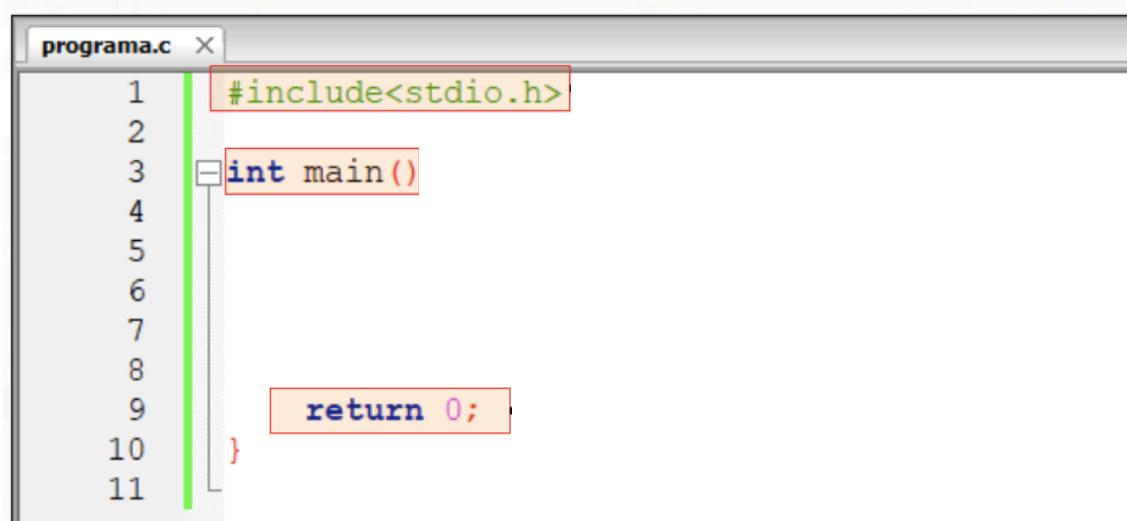
1 Introdução

2 Estrutura Básica em C

3 Entrada e Saída

4 Referências

# Estrutura básica em C



A screenshot of a code editor window titled "programa.c". The code editor has a light gray background and displays the following C program:

```
1 #include<stdio.h>
2
3 int main()
4 {
5
6
7
8
9     return 0;
10}
11
```

The code is syntax-highlighted: `#include`, `stdio.h`, `int`, `main`, `return`, and `0` are in blue; the brace at line 10 is in red; and the line numbers 1 through 11 are in green.

# Estrutura básica em C

A screenshot of a code editor window titled "programa.c". The code is as follows:

```
1 #include<stdio.h>
2
3 int main(){
4
5
6
7
8
9     return 0;
10}
11
```

The code is annotated with arrows pointing from specific lines to explanatory text:

- An arrow points from the line "#include<stdio.h>" to the text "Bibliotecas".
- An arrow points from the line "int main(){ " to the text "Função main".
- An arrow points from the line " return 0;" to the text "Retorno da função main".

A yellow callout box contains the text: "Indica o fim da execução da função main".

# Estrutura básica em C

The screenshot shows a code editor window titled "programa.c". The code is as follows:

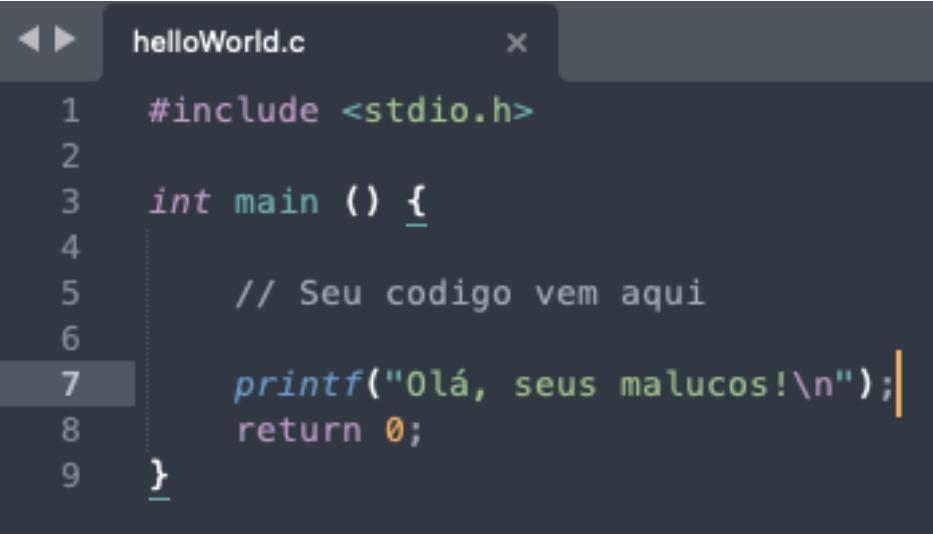
```
1 #include<stdio.h>
2
3 int main(){
4
5
6
7
8
9     return 0;
10}
11
```

Annotations explain parts of the code:

- #include<stdio.h> -> Bibliotecas
- int main() { -> Função main
- return 0; -> Retorno da função main
- Indica o fim da execução da função main

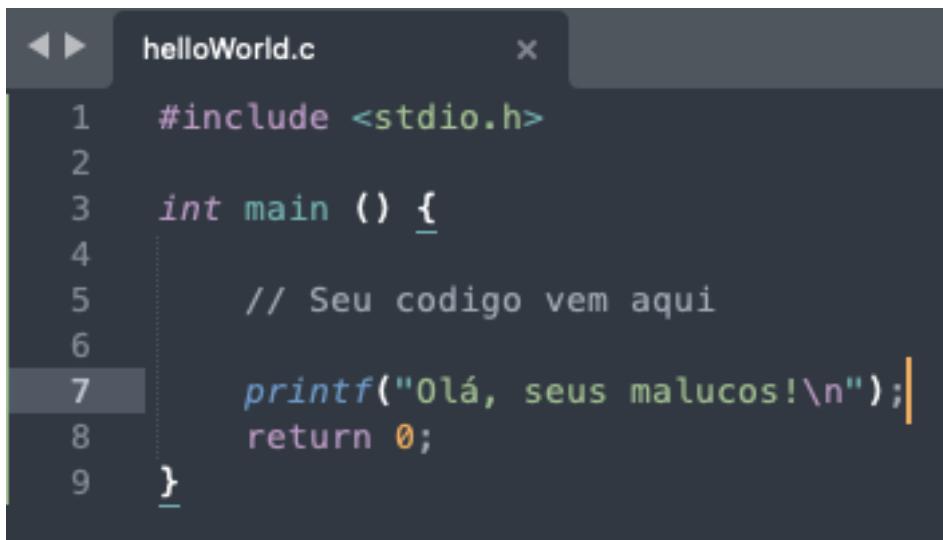
- **Bibliotecas** contém conjunto de funções e instruções previamente estabelecidas e que podem ser usadas pelo programa
- A execução do programa inicia-se pela função **main**

# Estrutura básica em C



```
helloWorld.c      x
1 #include <stdio.h>
2
3 int main () {
4
5     // Seu código vem aqui
6
7     printf("Olá, seus malucos!\n");
8     return 0;
9 }
```

# Estrutura básica em C



```
helloWorld.c      x
1 #include <stdio.h>
2
3 int main () {
4
5     // Seu código vem aqui
6
7     printf("Olá, seus malucos!\n");
8     return 0;
9 }
```

Saída para o usuário

>> Olá, seus malucos!  
>>

# IDE

- ***Integrated Development Environment (IDE)***
  - Interface gráfica
  - Facilita a edição do código-fonte
  - Facilita a compilação e análise de código
  - Facilita a depuração do código
  - Possibilita o desenvolvimento de projeto de software maiores



# Informações Gerais

- Uma variável sempre deve ser declarada antes de ser usada
  - Declarar indica ao compilador para reservar um espaço na memória para guardar o valor dessa variável

Exemplos de declaração

```
int main(){  
  
    int num1;  
    int num2;  
    float resultado;  
    float multiplicacao;  
    .  
    .  
    .
```



```
int main(){  
  
    int num1, num2;  
    float resultado, multiplicacao;  
    .  
    .  
    .
```

# Roteiro

- 1** Introdução
- 2** Estrutura Básica em C
- 3** Entrada e Saída
- 4** Referências

# Entrada e saída de dados



Entrada:

```
scanf(formato, &variável);
```

Exemplos:

```
scanf("%f", &n);
scanf("%d", &num);
scanf("%d", &cont);
```

Saída:

```
printf(conteúdo, [parâmetros]);
```

Exemplos:

```
printf("%f", n);
printf("Exame");
printf("O numero e' %d", num);
printf("Sao necessarias %d moedas", cont);
```

Ambas as funções scanf e printf pertencem a biblioteca stdio.h

# Saída de dados

Sintaxe geral do comando printf

```
printf("Mensagem escrita na tela", lista_argumentos);
```

Exemplos:

```
int num = 5;  
char letra = 'C';  
float pi = 3.1415;
```

```
printf("O valor de num e' %d", num);  
printf("O valor de num e' %d e o valor de letra e' %c", num, letra);  
printf("num = %d, letra = %c e pi = %f", num, letra, pi);
```

%c → caractere (char)  
%d ou %i → inteiro (int)  
%f → ponto flutuante (float)

# Saída de dados

- Códigos usados na função printf()

Código	Formato
%c	Caractere
%d ou %i	Inteiro decimal com sinal
%e	Notação científica
%f	Ponto flutuante decimal
%g	Menor representação entre %f e %e
%o	Octal sem sinal
%s	<i>String</i> de caracteres
%u	Inteiro sem sinal (unsigned int)
%x	Hexadecimal sem sinal
%%	Símbolo %

# Saída de dados

Caractere	Significado
\a	Caractere (invisível) de aviso sonoro (bip)
\n	Caractere (invisível) de nova linha
\t	Caractere (invisível) de tabulação horizontal
\\"	Caractere de barra invertida \'
\'	Caractere de aspas simples
\”	Caractere de aspas duplas
\?	Sinal de interrogação

Obs.: Não deve conter espaço entre \ e o símbolo desejado

# Saída de dados com acentos

```
#include<stdio.h>
#include<locale.h> //para usar a função setlocale()

int main(){
    printf("Apareça acentos: á, ê, õ ...\n");
    //Altera a codificação de saída para o padrão do S.O., em Português...

    setlocale(LC_ALL, " ");
    printf("Apareça acentos: á, ê, õ ...\n");

    return 0;
}
```

Observação: A alteração no Windows vale também para número! O padrão para casas decimais é mostrar com ponto “.”, e com o `setlocale()` mostrará com vírgula “,”

# Entrada de dados

- A função `scanf()` permite que o usuário forneça dados pelo teclado para o programa

```
scanf("expressão de controle", lista de argumentos);
```

Indica ao compilador o TIPO  
de dado está se esperando que  
o usuário digite

- “Expressão de controle”: “%formato”, parecido com o `printf...`
- Argumento: `&nome_variável`
  - Indica o endereço de memória onde se armazena o conteúdo de `nome_variável`

```
scanf("%i", &variavel_inteira);
scanf("%f", &variavel_float);
scanf("%c", &variavel_char);
```

# Entrada de dados

- Relembrando alguns aspectos sobre variáveis:
  - Possui um nome e tipo;
  - É associado a um endereço de memória;
  - Ao utilizar uma variável, acessa-se o seu conteúdo/valor localizado no endereço de memória associado a essa variável;
  - Ao atribuir um valor para uma variável, modifica-se o seu conteúdo/valor;
  - O **&** serve para obter o endereço de memória associado a uma variável;

```
int idade = 10;  
printf("Endereco de memoria associado a variavel idade: %p \n",&idade);  
printf("Valor armazenado por idade: %i \n", idade);  
scanf("%i", &idade); //recebe um int e armazena no end. de memória associado à variável idade  
printf("Endereco de memoria associado continua igual: %p \n",&idade);  
printf("Novo valor armazenado por idade: %i \n", idade);
```

# Exemplo de saída e entrada

```
#include<stdio.h>

int main(){
    int idade;

    printf("Digite sua idade: ");
    scanf("%i", &idade);

    printf("Sua idade é: %i anos", idade);
    printf("Daqui 5 anos você terá %i anos", idade + 5);

    return 0;
}
```

# Comentários

- Em regra, são utilizados para a documentação do código;
- Objetiva facilitar o entendimento do programa por outros programadores ou o próprio criador;
- Comentários na linguagem C podem ser feitos de duas formas:

`/* < código > */` Para um determinado trecho de código (várias linhas)

`// < código >` Para uma linha de código, sendo colocado em seu início

# Roteiro

- 1 Introdução**
- 2 Estrutura Básica em C**
- 3 Entrada e Saída**
- 4 Referências**

# Perguntas?

Prof. Rafael G. Mantovani

[rafaelmantovani@utfpr.edu.br](mailto:rafaelmantovani@utfpr.edu.br)

Prof. Adalberto Lazarini

[adalbertoz@utfpr.edu.br](mailto:adalbertoz@utfpr.edu.br)