

# Laboratório de Programação

Prof. Ms. Hugo Régis

Adaptado de:

Profa. Ms. Valéria Pinheiro



**UNIVERSIDADE  
FEDERAL DO CEARÁ**  
CAMPUS DE RUSSAS

# Aula 02 – Introdução à Linguagem C

# Linguagem C

- É uma linguagem de baixo / médio nível
- É uma linguagem imperativa.
- Permite que um problema complexo seja estruturado em problemas mais simples.

Desenvolvida na década de 70 por:



Dennis Ritchie

Cientista da computação



Ken Thompson

Cientista da computação

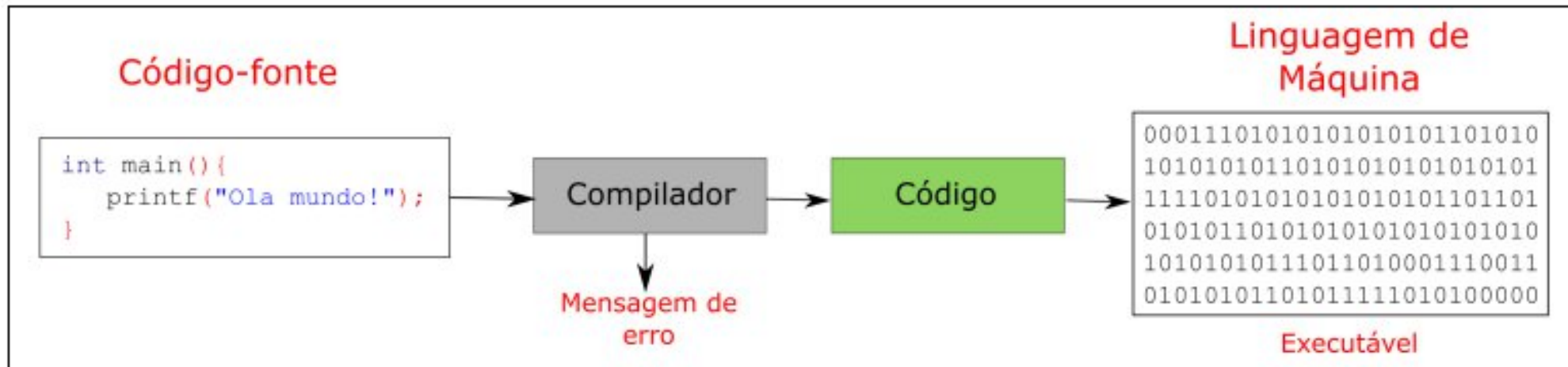
# Linguagem C

- Usada para desenvolver o sistema operacional UNIX, e hoje, base para novas linguagens, entre elas C++, C# e Java.
- Sintaxe estruturada, flexível e simples
- Compartilha recursos tanto de alto quanto de baixo nível
- Programas em C são compilados, gerando programas executáveis



# Programação em Linguagem C

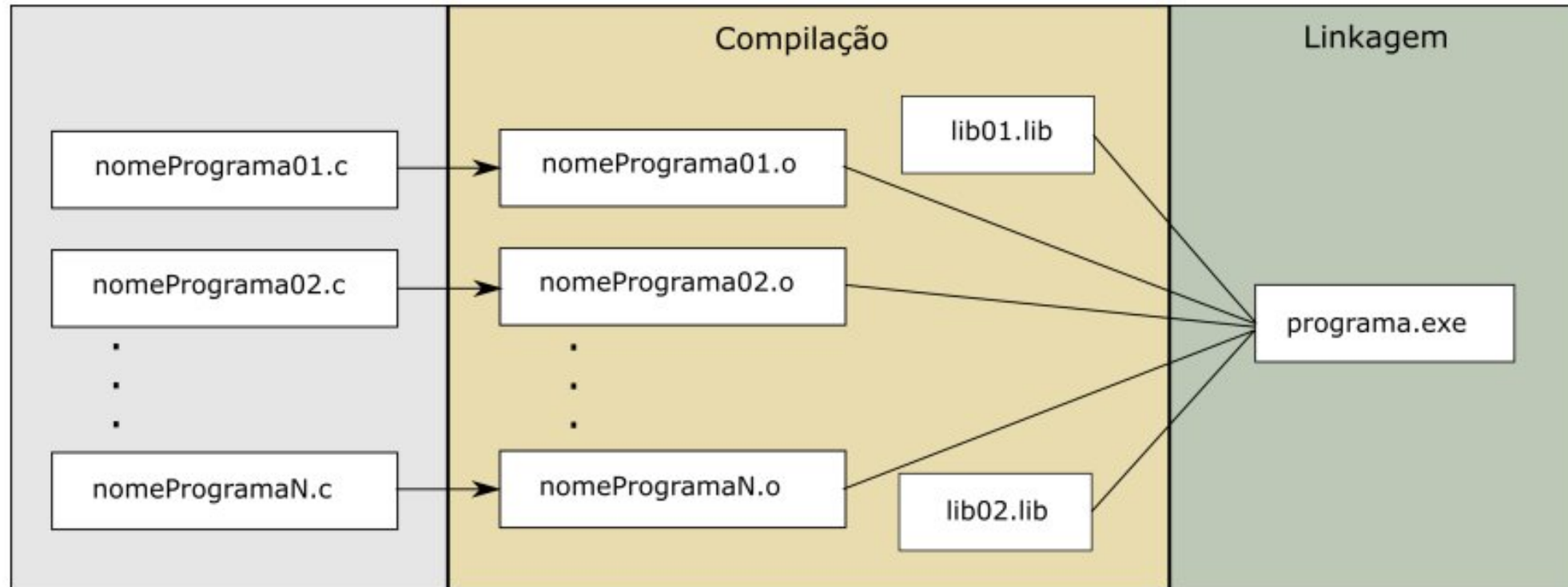
Etapas de construção de uma aplicação



Fonte: (DAMAS, 2007)

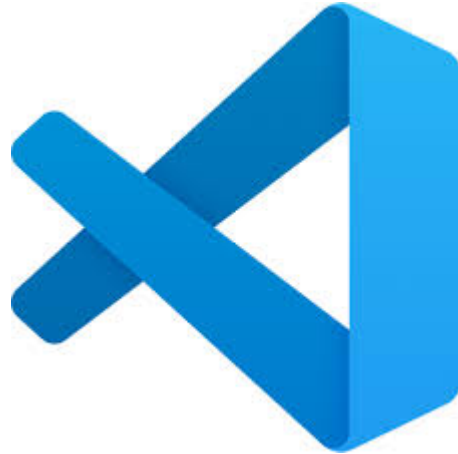
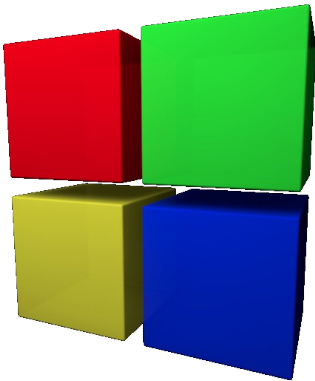
# Programação em Linguagem C

Processo de compilação e “linkagem”



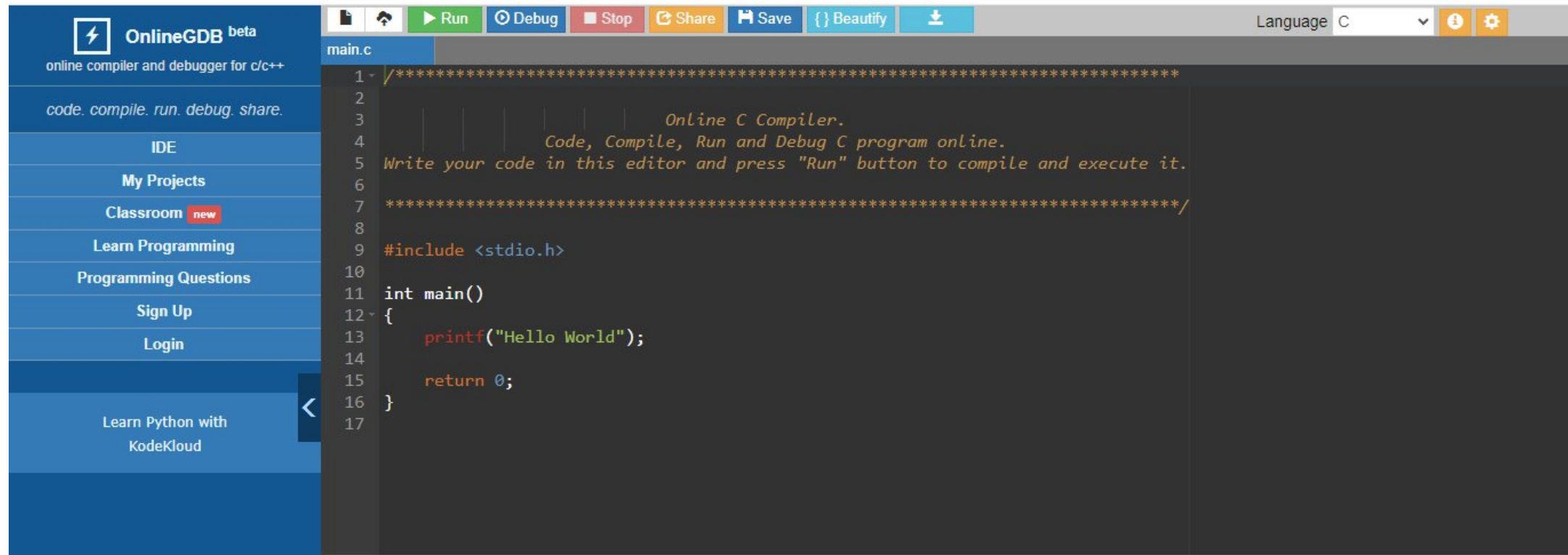
Fonte: (DAMAS, 2007)

# Compiladores / IDEs



# Compiladores / IDEs

[https://www.onlinegdb.com/online\\_c\\_compiler](https://www.onlinegdb.com/online_c_compiler)

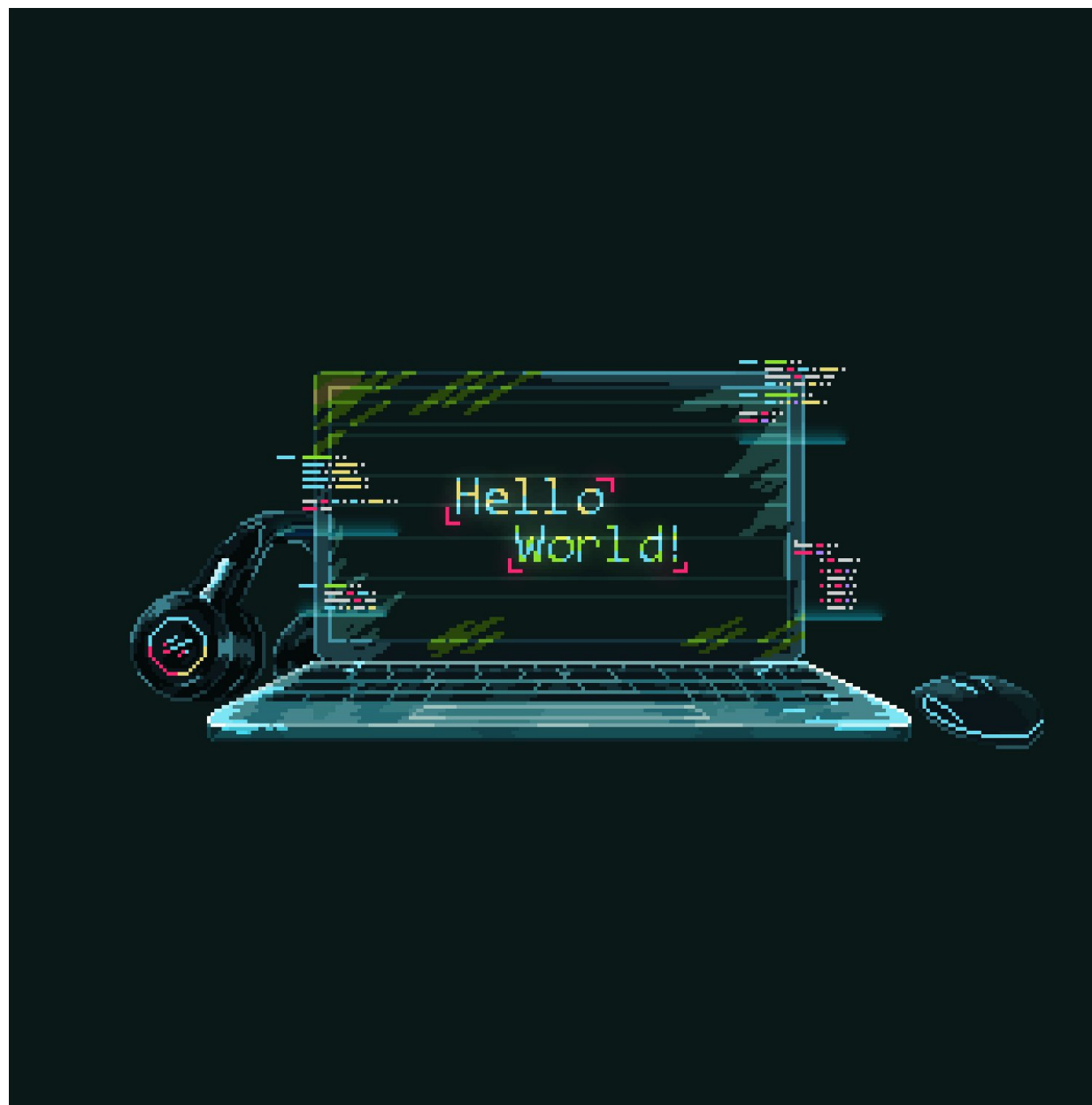


The screenshot displays the OnlineGDB web IDE interface. On the left is a blue sidebar with navigation links: OnlineGDB beta, code. compile. run. debug. share., IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, Login, and Learn Python with KodeKloud. The top of the editor features a toolbar with icons for file operations, a 'Run' button, 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. A 'Language' dropdown menu is set to 'C'. The main editor area, titled 'main.c', shows a C program with line numbers 1 through 17. The code includes a multi-line comment at the top and a 'Hello World' program structure.

```
1 - /*****  
2  
3  
4 | | | | | Online C Compiler.  
5 | | | | | Code, Compile, Run and Debug C program online.  
6 | | | | | Write your code in this editor and press "Run" button to compile and execute it.  
7 *****/  
8  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```



# Meu primeiro Programa em C



# Um primeiro programa em C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void main() {
4
5  }
```

<https://petbcc.ufscar.br/stdlib/>

<https://petbcc.ufscar.br/stdio/>

# Um primeiro programa em C

- Comandos de **impressão** (na tela)

```
1 //Sintaxe:  
2 printf("formato", argumentos);
```

- O comando ***printf***  
    printf("Ola mundo!");  
    printf("Ola mundo!\n");
- **imprime todos os caracteres** que estão entre aspas

# Um primeiro programa em C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void main() {
4      printf("Ola Mundo!");
5  }
```

# ATENÇÃO !

Preste **MUITA** atenção ao digitar seu programa!

É muito fácil **ESQUECER UM PONTO E VÍRGULA** ou de fechar chaves ou parênteses.

# Compilando o seu programa

```
~/Área de Trabalho$ gcc hello.c -o hello  
~/Área de Trabalho$
```



Programa compilado com sucesso ;-)

# Compilando o seu programa

```
mariavm@ubuntu:~$ cd Área\ de\ Trabalho/  
mariavm@ubuntu:~/Área de Trabalho$ gcc hello.c -o hello  
hello.c: Na função 'main':  
hello.c:12:2: erro: expected ';' before 'return'  
mariavm@ubuntu:~/Área de Trabalho$
```



# Execute o seu programa

```
mariavm@ubuntu:~/Área de Trabalho$ ./hello
```

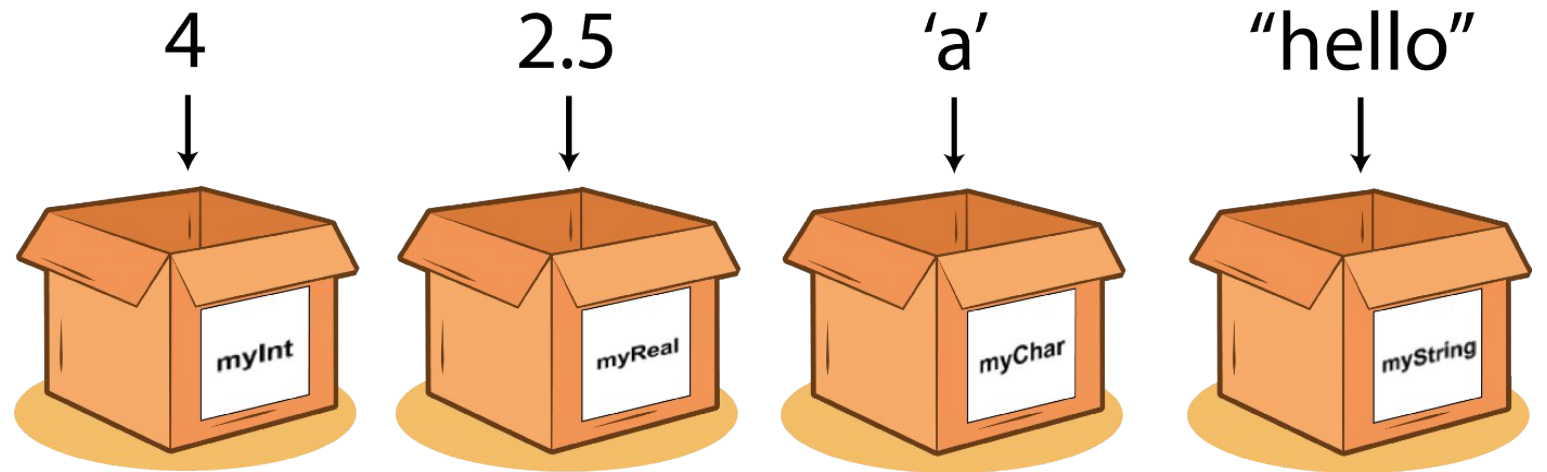




# Exercícios

- 1**-Fazer um programa que imprima o nome da sua cidade natal.
- 2**-Faça um programa que imprima na primeira linha seu nome, na segunda sua idade e na terceira sua altura.

# Variáveis



# Variáveis

- Todo programa precisa de um **espaço na memória** para trabalhar.
- As **declarações de variáveis** reservam o espaço necessário para isso.
- Em C, todas as variáveis precisam ser **declaradas no início** de cada bloco de comandos
- Para declarar uma variável é necessário definir o seu **tipo**.

# Tipos em C

<i>Tipo</i>	<i>Descrição</i>
<b>int</b>	Utilizado para definir uma variável inteira que comporta valores pertencentes ao conjunto dos números inteiros.
<b>long int</b>	Utilizado para definir uma variável inteira que comporta valores maiores que o <b>int</b> .
<b>short int</b>	Utilizado para definir uma variável inteira que comporta valores menores que o <b>int</b> .
<b>float</b>	Utilizado para definir uma variável real que comporta valores pertencentes ao conjunto dos números reais.
<b>double</b>	O tipo <b>double</b> é similar ao <b>float</b> , a diferença é que o este tipo comporta valores reais com um número de dígitos maior, assim, a precisão nos cálculos com casas decimais aumenta.
<b>char</b>	Utilizado para armazenar um caractere. Comporta apenas um caractere.

# Declaração de Variáveis

- Para declarar uma **variável inteira** em C, fazemos:  
**tipo-da-variável** <nome-da-variável>;

```
1 int idade;  
2 float peso;  
3 char genero;  
4 double rendimento;
```

A linguagem C é "*case sensitive*", ou seja, letras maiúsculas e minúsculas correspondentes são consideradas diferentes

# Regras para a definição de variáveis

<i>Forma incorreta</i>	<i>Motivo</i>	<i>Sugestão de correção</i>
<b>float</b> 4nota;	Não é permitido iniciar o nome da variável com números.	<b>float</b> nota4;
<b>char</b> float;	Não é permitido utilizar palavra reservada como nome de uma variável.	<b>char</b> vFloat;
<b>double</b> vinte%;	Não é permitido utilizar os seguintes caracteres especiais como %, @, #, \$, &, etc.	<b>double</b> vintePercent;
<b>int</b> idade pes;	Não é permitido separar os nomes compostos em variáveis.	<b>int</b> idade_pes;

# Palavras reservadas

---

<i>Palavras Reservadas</i>					
<b>auto</b>	<b>break</b>	<b>case</b>	<b>char</b>	<b>const</b>	<b>continue</b>
<b>default</b>	<b>do</b>	<b>double</b>	<b>else</b>	<b>enum</b>	<b>extern</b>
<b>float</b>	<b>for</b>	<b>goto</b>	<b>if</b>	<b>int</b>	<b>long</b>
<b>main</b>	<b>register</b>	<b>return</b>	<b>short</b>	<b>signed</b>	<b>sizeof</b>
<b>static</b>	<b>struct</b>	<b>switch</b>	<b>typedef</b>	<b>union</b>	<b>unsigned</b>
<b>void</b>	<b>volatile</b>	<b>while</b>			

---

Fonte: Adaptado de [Schildt \(1996, p. 10\)](#)

# Atribuição de Variáveis

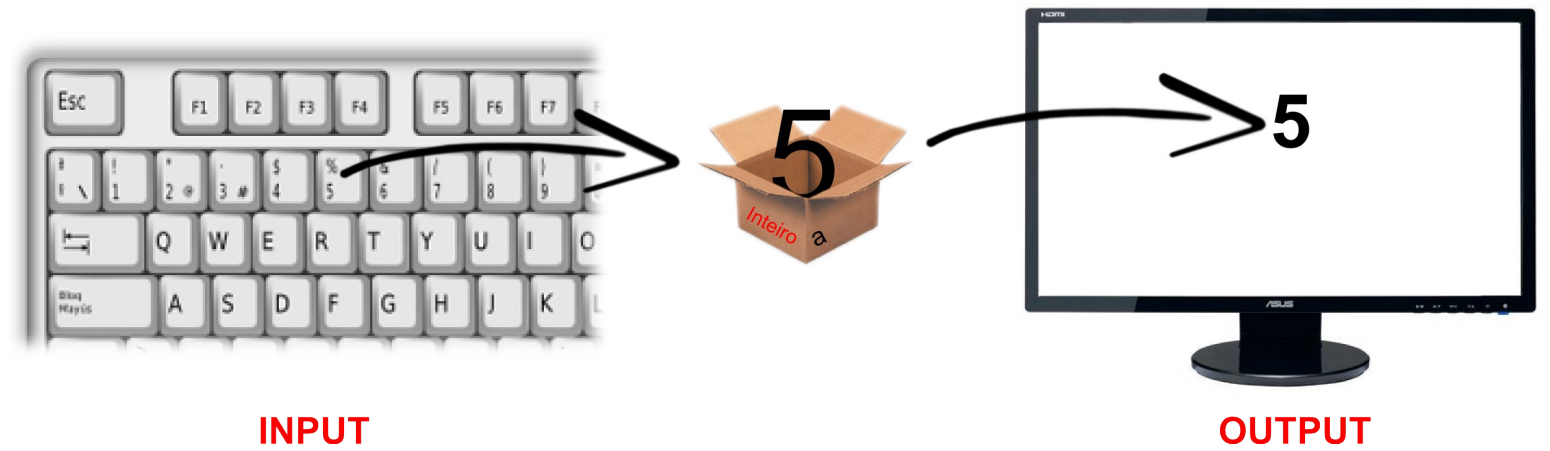
- Para atribuir uma **variável inteira** em C, utilizamos o sinal de igualdade “=”:

**tipo-da-variável** <nome-da-variável> = valor;

```
1  int idade, matricula;  
2  idade = 30;  
3  matricula = 123659;  
4  float preco = 42.9;
```



# Entrada e Saída



# Função printf()

- Comandos de **impressão** (na tela)

```
1 //Sintaxe:  
2 printf("formato", argumentos);
```

- O comando ***printf***  
    printf("Ola mundo!");  
    printf("Ola mundo!\n");
- **imprime todos os caracteres** que estão entre aspas

# Função printf()

- Comandos de **impressão** (na tela)

```
1 float total = 300 + 400;  
2 printf("Total da conta: %f", total);
```

# Função printf()

- Formatadores de tipos em C

<i>Formato</i>	<i>Tipo da variável</i>	<i>Conversão realizada</i>
<b>%c</b>	Caracteres	char, short int, int, long int
<b>%d</b>	Inteiros	int, short int, long int
<b>%e</b>	Ponto flutuante, notação científica	float, double
<b>%f</b>	Ponto flutuante, notação decimal	float, double
<b>%lf</b>	Ponto flutuante, notação decimal	double
<b>%g</b>	O mais curto de %e ou %f	float, double
<b>%o</b>	Saída em octal	int, short int, long int, char
<b>%s</b>	String	char *, char[]
<b>%u</b>	Inteiro sem sinal	unsigned int, unsigned short int, unsigned long int
<b>%x</b>	Saída em hexadecimal (0 a f)	int, short int, long int, char
<b>%X</b>	Saída em hexadecimal (0 a F)	int, short int, long int, char
<b>%ld</b>	Saída em decimal longo	Usado quando long int e int possuem tamanhos diferentes.

# Função printf()

- Para formatar um float

```
1 \ %[tam].[casa_dec]f
```

Em que:

- **tam** – indica o tamanho mínimo que deve ser impresso na saída. Se o número possuir um tamanho superior ao informado neste parâmetro, o número não será truncado.
- **casa\_dec** – número de casas decimais que devem ser impressas. Neste caso, se o número possuir uma quantidade de casas decimais superior ao indicado, então as casas decimais serão truncadas.

# Função printf()

- Para formatar um float

```
1 \ %[tam].[casa_dec]f
```

```
1 float total = 300 + 400;  
2 printf("Total da conta: %3.2f", total);
```

SAÍDA DO PROGRAMA

TOTAL DA CONTA: 700.00

# Função scanf()

Comando de leitura pelo teclado

- O comando scanf

```
1 //Sintaxe:  
2     scanf("formato", enderecosArgumentos);
```

# Função scanf()

Comando de leitura pelo teclado

- O comando scanf

```
1 int mat;  
2 scanf("%d", &mat);
```

```
1 float nota1, nota2;  
2 scanf("%f %f", &nota1, &nota2);
```

- **%d** indicativo do tipo, neste caso do tipo inteiro.
- **&mat** operador utilizado para obter o endereço de memória da variável.
- espera o usuário digitar um número inteiro pelo teclado.
- após o usuário digitar a tecla <ENTER>, armazena o número digitado na variável



# Exemplo 1

Faça um programa em C que receba dois números inteiros e ao final imprima a soma deles.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void main() {
4      int soma, num1, num2;
5      printf("Informe o primeiro numero:");
6      scanf("%d", &num1);
7      printf("Informe o segundo numero:");
8      scanf("%d", &num2);
9
10     soma = num1 + num2;
11
12     printf("Resultado da soma: %d", soma);
13 }
```

## Exemplo 2

Escreva um programa em C que faz uma pergunta ao usuário

*quantos anos você tem?*

espera que o usuário entre com uma resposta numérica através do teclado e finaliza com um comentário sobre a idade do usuário.

## Exemplo 2

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

int main(){
    int idade;
    printf("Digite sua idade:\n");
    scanf("%d", &idade);
    printf("Sua idade: %d", idade);
    return 0;
}
```

# Atividade

Escreva um algoritmo em C que, lê o ano atual (Ex: 2025), lê o ano (Ex: 2000), e informa quantos anos o usuário tem ou fará no ano atual informado.

Exemplo:

Informe o ano em que estamos: 2025.

Informe o ano em que você nasceu: 2000.

Neste ano de 2025, você tem (ou fará) 25 anos.

Próxima aula

Tipos