

# PROGRAMAÇÃO COMPUTACIONAL PARA ENGENHARIA

## TIPOS DE DADOS

Maurício Moreira Neto<sup>1</sup>

<sup>1</sup> **Universidade Federal do Ceará**  
**Departamento de Computação**

30 de janeiro de 2020

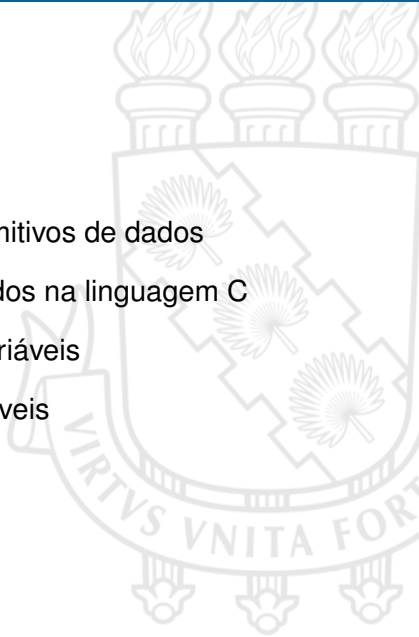
# Sumário

- 1 Objetivos
- 2 Linguagem de Programação
- 3 Linguagem C
- 4 Variáveis e Constantes
- 5 Escopo de Variáveis
- 6 Tipos Primitivos de Dados



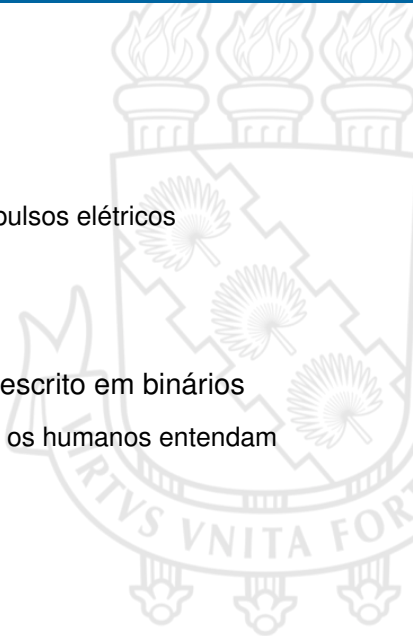
# Objetivos

- Aprender quais são os tipos primitivos de dados
- Quais as representações de dados na linguagem C
- Aprender sobre constantes e variáveis
- Aprender sobre escopo de variáveis



# Linguagem de Programação

- Linguagem de Máquina
  - Computador entende apenas pulsos elétricos
  - Presença ou não de pulso
  - 0 ou 1
- Tudo no computador dever ser descrito em binários
  - Porém, é muito difícil para que os humanos entendam códigos binários
  - 00011110



# Linguagem de Programação

## ■ Linguagem Assembly

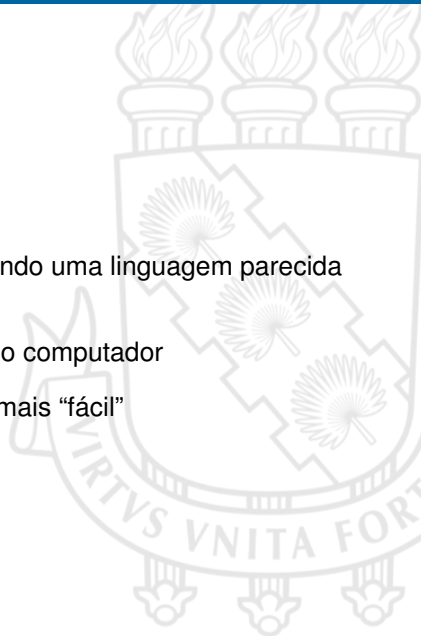
- Utiliza mnemônicos
- O conjunto de 0 e 1 são representados por um código
- 10011011 -> ADD

## ■ Os Problemas da linguagem Assembly

- Requer programação especial (assembly)
- Conjunto de instruções varia com o computador (processador)
- Programar ainda continua complexo com esta linguagem!

# Linguagem de Programação

- Linguagem de Alto Nível
  - Programas são escritos utilizando uma linguagem parecida com a linguagem humana
  - Independente da arquitetura do computador
  - A programação se torna algo mais “fácil”
  - Uso de compiladores



# Linguagem de Programação

## ■ Primórdios

- Uso da computação para cálculos de fórmulas
- As fórmulas eram traduzidas para a linguagem de máquinas
- **Por que não escrever programas parecidos com as fórmulas que se deseja computar?**

# Linguagem de Programação

- FORTRAN (FORmula TRANSform)
  - Em 1950, um grupo de programadores da IBM liderados por John Backus produz a versão inicial da linguagem
  - Primeira linguagem de alto nível
- Várias outras linguagens de alto nível foram criadas
  - Algol-60, Cobol, Pascal, etc



# Linguagem C

- Uma das mais bem sucedidas foi uma linguagem chamada C
  - Criada em 1972 nos laboratórios por Dennis Ritchie
  - Revisada e padronizada pela ANSI em 1989
    - ANSI: American National Standards Institute
    - Padrão mais utilizado



# Linguagem C

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

int main() {

    printf("Hello World \n");

    system("pause");

    return 0;

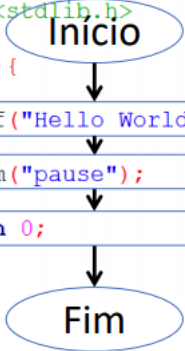
}
```



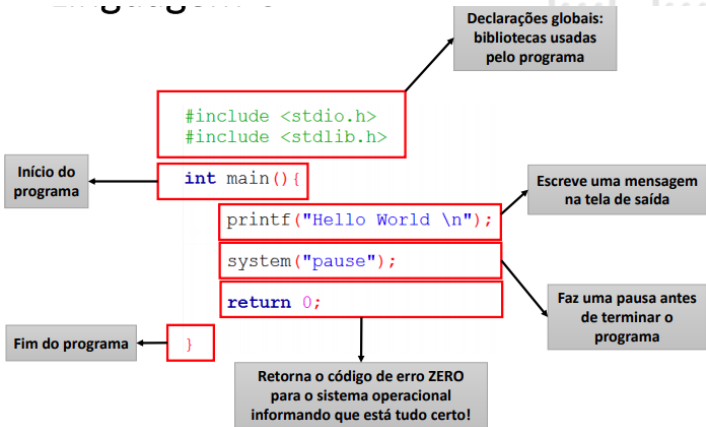
# Linguagem C

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

int main() {
    printf("Hello World \n");
    system("pause");
    return 0;
}
```



# Linguagem C



# Linguagem C

- Os comentários permitem adicionar uma descrição sobre o programa!
  - São ignorados pelo compilador

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

int main() {
    /*
     * A função printf serve
     * para escrever na tela
     */
    printf("Hello World \n");

    //faz uma pausa no programa
    system("pause");

    return 0;
}
```

## Variáveis e Constantes

# Variáveis e Constantes



# Variáveis

## ■ Matemática

- É uma entidade capaz de representar um valor ou expressão
- Pode representar um número ou um conjunto de números
- $f(x) = x^2$

## ■ Computação

- Posição de memória que armazena uma informação
- Pode ser modificada pelo programa
- **Deve ser definida antes de ser utilizada!**

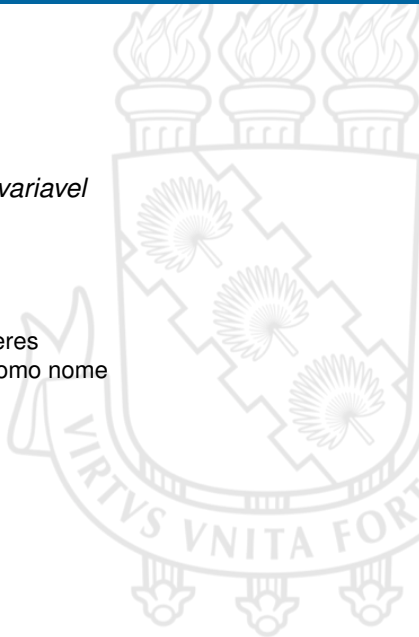
# Declarando Variáveis

- Precisa-se informar ao programa quais os dados queremos armazenar
- Precisa-se, também, informar o que são esses dados (de que tipo, faixa, ...)
  - Um nome de uma pessoa
    - Cadeira de caracteres (“Maurício” – 8 caracteres)
  - O valor da temperatura atual
    - Um valor numérico (com casas decimais)
  - Se uma cadeira no cinema está ou não vago
    - Um valor lógico (ocupado: verdadeiro / desocupado: falso)



# Declarando Variáveis

- Declaração de variáveis em C
  - `< tipo_de_dado > nome_da_variavel`
- Propriedades
  - Nome
    - Pode ter um ou mais caracteres
    - Nem tudo pode ser usado como nome
  - Tipo
    - Conjunto de valores aceitos
  - Escopo
    - Global ou local



# Declarando Variáveis

## ■ Identificador (Nome):

- 1 Deve ser iniciado com letras ou underscore (\\_)
- 2 Caracteres devem ser letras, números ou underscore
- 3 Palavras chaves não podem ser usadas como nomes
- 4 Letras maiúsculas e minúsculas são consideradas diferentes

## ■ Não utilizar:

- 1 Não utilizar espaços nos nomes. Ex.: nome do aluno
- 2 Não utilizar acentos ou símbolos. Ex.: garça, número, ...
- 3 Não inicializar o nome da variável com números. Ex.: 1<sup>a</sup>,52, ...
- 4 **Não pode haver duas variáveis com o mesmo nome!**

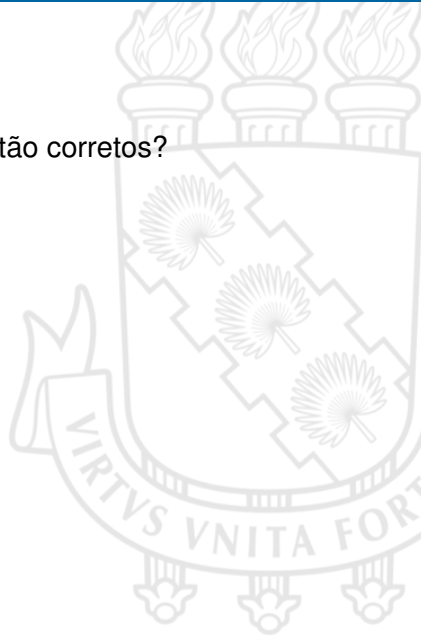
# Declarando Variáveis

- Não deve-se utilizar palavras-chaves como variáveis!
- Lista de palavras-chaves

auto	break	case	char	const	continue	do	double
else	for	int	union	static	default	void	return
enum	goto	long	unsigned	struct	extern	while	sizeof
float	if	short	volatile	switch	register	typeof	

# Declarando Variáveis

- Quais os nomes de variáveis estão corretos?
  - Contador
  - contador1
  - comp!
  - .var
  - Teste\_123
  - \_teste
  - int
  - int1
  - 1contador
  - -x
  - Teste-123
  - x&



# Declarando Variáveis

- Os corretos são:
  - **Contador**
  - **contador1**
  - comp!
  - .var
  - **Teste\_123**
  - **\_teste**
  - int
  - **int1**
  - 1contador
  - -x
  - Teste-123
  - x&



# Constantes

- Como uma variável, uma constante também armazena um valor na memória do computador
- Entretanto, esse valor não pode ser alterado: **pois é constante!**
- **Para constantes, é obrigatório a atribuição de um valor!**

# Constantes

## ■ Usando **#define** (Macro)

- Deve-se incluir a diretiva de pré-processamento
- **#define** antes do início do código:
- **CUIDADO: nesse caso, não deve-se colocar o “;”**
- **Exemplo:** #define PI 3.1415

## ■ Usando **const**

- Usando o **const**, a declaração não precisa estar no início do código
- A declaração é igual a de uma variável inicializada
- **Exemplo:** const double pi = 3.1415

# Enumerações

- As enumerações são listas de constantes ou *constantes enumeradas*
- Utiliza-se a palavra **enum** em conjunto com uma lista de constantes enumeradas separadas por vírgula
- Cada constante recebe um valor padrão ou um valor atribuído
  - O primeiro identificador assume o valor atribuído ou **o valor zero, caso não seja definido**
  - Os demais assumem os valores atribuídos ou **o valor anterior incrementado + 1**



# Enumerações

```
enum <etiqueta> {lista_de_identificadores};
```

## ■ Exemplos de Declaração

```
enum semana {seg, ter, qua, qui, sex};  
enum semana dia;
```

**OU**

```
enum semana {seg, ter, qua, qui, sex} dia;
```

# Enumerações

## ■ Exemplo de Código

```
#include <stdio.h>
enum numero {zero, um, dois, tres, quatro, cinco};
int main(void) {
enum numero num;
num = zero;
printf("O número da variável num é: %d", num);
return 0;
}
```

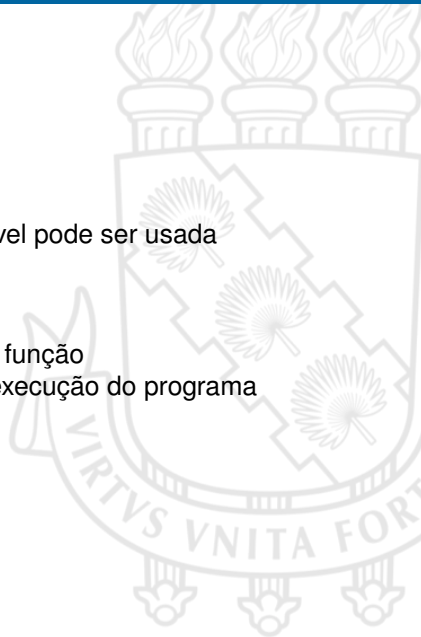
## Escopo de Variáveis

# Escopo de Variáveis



# Escopo de Variáveis

- Escopo
  - Define onde e quando a variável pode ser usada
- Escopo Global
  - Fora de qualquer definição de função
  - Tempo de vida é o tempo de execução do programa
- Escopo Local
  - Bloco ou função

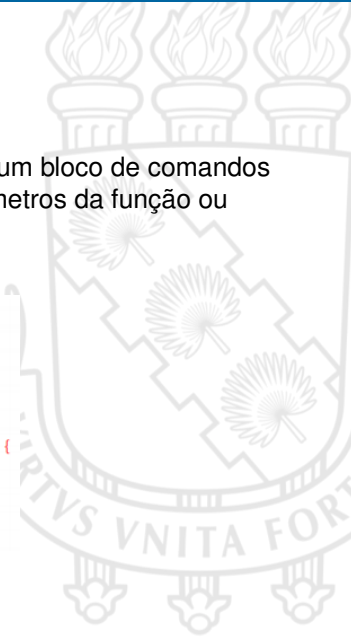


# Escopo de Variáveis

## ■ Escopo Local

- Bloco: visível apenas no interior de um bloco de comandos
- Função: declarada na lista de parâmetros da função ou definida dentro da função

```
//bloco
if(x == 10) {
    int i;
    i = i + 1;
}
//função
int soma (int x, int y) {
    int z;
    z = x + y;
    return z;
}
```



# Escopo de Variáveis

```

#include <stdio.h>
#include <stdlib.h>
void func1(){
    int x;//variável local
}
void func2(){
    int x;//variável local
}
int main(){
    int x;
    scanf("%d",&x);
    if(x == 5){
        int y=1;
        printf("%d\n",y);
    }
    system("pause");
    return 0;
}

```

- Escopo global
- Escopo local
- Escopo local dentro de outro escopo local



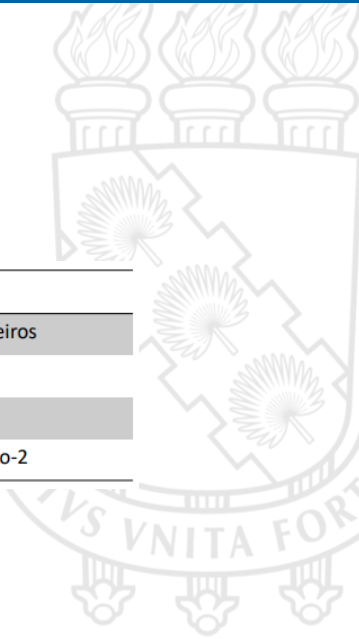
# Tipos de Dados

- Um tipo de dado especifica um conjunto de valores determinando sua natureza, seu tamanho, sua representação e sua imagem
- **Natureza:** caracteriza o tipo representado
  - **Ex:** caractere, número inteiro, um número real, ...
- **Tamanho:** Determina o tamanho em bits necessário para armazenar os valores do tipo
- **Representação:** Determina a forma como os bits armazenados devem ser interpretados
- **Imagem:** Determina a faixa de valores para o tipo

# Tipos de Dados

■ **Exemplo:** tipo de dado *tipo\_exem*

<b>Especificador</b>	<b>tipo_exem</b>
Natureza	Números inteiros
Tamanho	8 bits
Imagem	[-128; 127]
Representação	Complemento-2



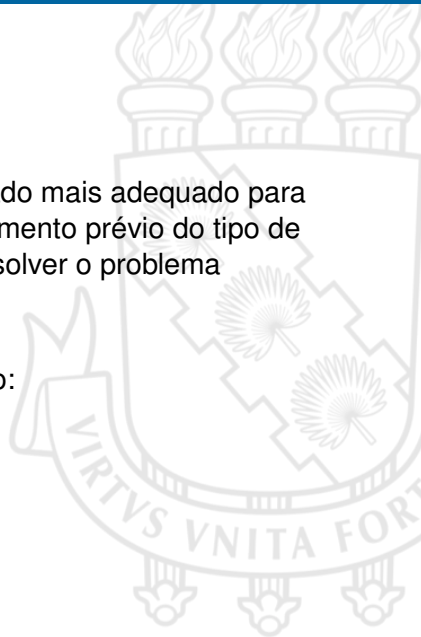


# Tipos de Dados

- As expressões usadas para identificar um tipo de dados é chamado de **especificador de tipo**
  - Normalmente consiste em um ou mais nomes
  - **Exemplo:** *short* e *short int* designam o mesmo tipo
- Um dado pode ter várias representações e uma mesma representação pode ser implementadas de diferentes tamanhos

# Tipos Primitivos de Dados

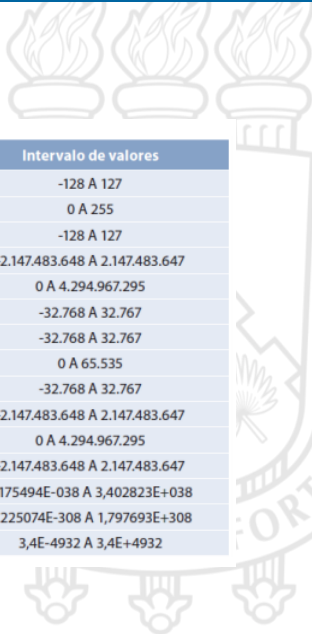
- É importante definir o tipo de dado mais adequado para ser armazenado e ter o conhecimento prévio do tipo de informação a ser usado para resolver o problema
- Os tipos primitivos de dados são:
  - Literal
  - Lógicos
  - Numéricos



# Tipos Primitivos de Dados

- **Literais:** Recebe um caractere ou uma sequência de caracteres que podem ser letras, dígitos e símbolos especiais
- **Numéricos:** Podem ser divididos em dois tipos (Inteiros e Reais)
  - **Inteiros:** Recebe números inteiros positivos ou negativos
  - **Reais:** Recebe números reais, ou seja, permite o uso de casas decimais positivas ou negativas
- **Lógicos:** Recebe verdadeiro (1) ou falso (0)

# Tipos Primitivos de Dados



	Tipo	Bits	Intervalo de valores
<b>Literais</b>	char	8	-128 A 127
	unsigned char	8	0 A 255
	signed char	8	-128 A 127
<b>Numéricos</b>	int	32	-2.147.483.648 A 2.147.483.647
	unsigned int	32	0 A 4.294.967.295
	signed int	32	-32.768 A 32.767
	short int	16	-32.768 A 32.767
	unsigned short int	16	0 A 65.535
	signed short int	16	-32.768 A 32.767
	long int	32	-2.147.483.648 A 2.147.483.647
	unsigned long int	32	0 A 4.294.967.295
	signed long int	32	-2.147.483.648 A 2.147.483.647
	float	32	1,175494E-038 A 3,402823E+038
	double	64	2,225074E-308 A 1,797693E+308
	long double	96	3,4E-4932 A 3,4E+4932

# Tipos Primitivos de Dados

- **char:** um byte que armazena o código de um caractere do conjunto de caracteres local
  - Caracteres sempre ficam entre 'aspas simples'!

```
char UnidadeTemperatura; //pode ser 'C' ou 'F'  
char genero; //pode ser 'M' ou 'F'  
char opcoes; //pode ser '1', '2', ...
```

- **int:** um inteiro cujo tamanho depende do processador, tipicamente 16 ou 32 bits

```
int numeroAlunos;  
int idade;  
int numeroContaCorrente;  
int x = 10; //variavel x recebe o valor 10
```

# Tipos Primitivos de Dados

## ■ Números reais:

- Tipos: float, double e long double
- A parte decimal usa **ponto** e **não vírgula!**
- **float**: um número real com precisão simples
- **Double**: um número real com precisão dupla

```
float temperatura; //ex: 33.3  
float mediaNotas; //ex: 7.9  
float tempoTotal; //ex: 0.000032
```

- **Double**: um número real com precisão dupla
  - Números muito grandes ou muito pequenas

```
double distanciaGalaxias;  
double massaMolecular  
double totalEmpresa;
```

# Tipos Primitivos de Dados

- Número reais
  - Pode-se escrever números reais usando notação científica

```
double tempoTotal = 0.00000434;  
// notacao cientifica  
double tempoTotal = 3.134e-009;
```

# Tipos Primitivos de Dados

## ■ Tipos Booleanos

- Um tipo booleano pode assumir dois valores:
  - Verdadeiro ou falso (true ou false)
- Na linguagem C não existe o tipo de dado booleano. Para armazenar esse tipo de informação, use-se uma variável do tipo **int** (número inteiro)
  - **Valor 0 significa falso**
  - **Números + ou - : verdadeiro**

```
int assentoVago = 1; // verdadeiro
int portaAberta = 0; // falso
```



## Referências

- André Luiz Villar Forbellone, Henri Frederico Eberspächer, **Lógica de programação** (terceira edição), Pearson, 2005, ISBN 9788576050247.
- Ulysses de Oliveira, **Programando em C - Volume I - Fundamentos**, editora Ciência Moderna, 2008, ISBN 9788573936599
- **Slides baseados no material do site “Linguagem C Descomplicado”**
  - <https://programacaodescomplicada.wordpress.com/complementar/>