

Cláudio C. Rodrigues  
Faculdade da Computação - UFU

# ◦ **LINGUAGEM C - 01: TIPOS, VARIÁVEIS E E/S PADRÃO**

## A Linguagem C

- Linguagem de alto nível, genérica. Foi desenvolvida por programadores para programadores.
  - tendo como meta características de flexibilidade e portabilidade.
- Nasceu juntamente com o advento da teoria de linguagem estruturada e do computador pessoal.
- Usada para desenvolver o sistema operacional UNIX, e hoje, base p/ novas linguagens, entre elas a linguagem **C++** e **Java**.

## 1.1 Características

- **C** é uma linguagem de alto nível com uma sintaxe bastante estruturada e flexível tornando sua programação bastante simplificada.
- Programas em **C** são compilados, gerando programas executáveis.
- **C** compartilha recursos tanto de alto quanto de baixo nível, pois permite acesso e programação direta do microprocessador.

## 1.1 Características

- O compilador **C** gera códigos mais enxutos e velozes do que muitas outras linguagens.
- Embora estruturalmente simples (poucas funções intrínsecas), **C** não perde funcionalidade pois permite a inclusão de uma grande quantidade de *funções/procedimentos* do usuário.
  - Os fabricantes de compiladores fornecem uma ampla variedade de rotinas pré-compiladas disponibilizadas em **bibliotecas**.

## 1.1 Características

- Combina elementos de linguagens de alto nível (*Basic*, *Pascal*, ...) com a funcionalidade da linguagem de baixo nível (*Assembler*, ...)
- Flexível, Portátil e eficiente
- Não efetua nenhuma verificação em tempo de execução
  - (responsabilidade do programador)
- O padrão **ANSI** implementa o conceito de protótipo de função.

## 1.2 Histórico

- Origem da **Linguagem C**
  - A linguagem **C** foi inventada e implementada primeiramente no sistema operacional UNIX em um DEC PDP-11 por *Dennis M. Ritchie* nos laboratórios Bell.
  - **C** é o resultado do processo evolutivo que começou com a linguagem BCPL, que influenciou na elaboração da linguagem B de *Ken Thompson*.
  - Com a popularidade dos microcomputadores surgem muitas implementações de **C** com algumas discrepâncias.
  - A *American National Standards Institute* (**ANSI**) estabelece um padrão na década de oitenta.
    - Como consequência, grandes empresas de software preocuparam-se em atender o padrão ANSI.

## 1.3 Estrutura de um programa C

```
#include <bibliotecas>
#define <constantes>

protótipos_funções;
declaração_variáveis_globais;
declaração_funções;

int main()
{
    definição_variáveis_locais;
    comandos;
    return 0;
}
```

## 1.3 Estrutura dos Programas

- A função é o principal componente estrutural em **C**
- Programa em **C** consiste em uma ou várias funções
  - (os termos *programa* e *função* se confundem em **C**)
- A *função* pode ter qualquer nome (*identificador*), mas a *função main* é obrigatória, pois é a partir dela que se inicia a execução de um programa em **C**.

# Programa exemplo 1

- // Calcular o dobro de um valor

```
#include <stdio.h>

int main() {
    int n1, n2;
    printf("Entre um número:");
    scanf("%d", &n1);
    n2 = n1 * 2;
    printf("o dobro do número %d é: %d", n1, n2);
    return 0;
}
```

## 1.4 Normas Gerais: Caracteres Válidos

- Um programa-fonte em **C** é um texto não formatado escrito em um editor de textos usando um o conjunto padrão de caracteres **ASCII**.
- Abaixo estão os caracteres utilizados em **C**:

a b c d e f g h i j k l m n o p q r s t u v w x y z  
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
1 2 3 4 5 6 7 8 9 0  
+ - \* / \ = | & ! ? # % ( ) { } [ ] \_ ' " . , : < >

## 1.4 Normas Gerais (cont.)

- **Comentários:**

- podem ser escritos em qualquer lugar do texto.
- Para que o comentário seja identificado como tal, ele deve ter um `/*` antes e um `*/` depois.

- **Exemplo:**

`/*` esta é uma linha de comentário em C `*/`

`//` este é outro comentário

- válido apenas em **IDEs C++**

## 1.4 Normas Gerais (cont.)

- **Diretivas de Compilação:**

- são comandos que são processados durante a compilação do programa.
- informam ao compilador **C** quais são as constantes simbólicas usadas no programa e quais bibliotecas devem ser anexadas ao programa executável.



## 1.4 Normas Gerais (cont.)

- A diretiva **#include**

- diz ao compilador para incluir na compilação do programa outros arquivos.
- Geralmente estes arquivos contem funções de bibliotecas ou rotinas do usuário.

Arquivo	Descrição
stdio.h	Funções de entrada e saída (I/O)
stdlib.h	Funções de uso genérico
string.h	Funções de tratamento de strings
math.h	Funções matemáticas
ctype.h	Funções de teste e tratamento de caracteres

## 1.4 Normas Gerais (cont.)

- A diretiva **#define**

- diz ao compilador quais são as constantes simbólicas usadas no programa.

**#define** NOME\_CONST valor\_associado

- Onde

- **#define** é uma diretiva de compilação que diz ao compilador para trocar as ocorrências de NOME\_CONST pelo valor\_associado.

- Exemplos:

- #define PI 3.1415926536
- #define ON 1
- #define OFF 0

O uso da diretiva **#define** não se restringe apenas a definição de constantes simbólicas. Ela também pode ser usada para definir **macro instruções**.

## Programa exemplo 2

- // Programa para calcular a área de um círculo

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

```
#define PI 3.14159
```

```
int main() {
```

```
    double raio = 0.0, area;
```

```
    printf("Digite valor do raio\n");
```

```
    scanf("%f", &raio);
```

```
    area = PI * raio * raio;
```

```
    printf("A área do círculo de raio %f eh %f", raio, area);
```

```
    return 0;
```

```
}
```

## 1.5 Palavras Reservadas

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



# Palavras-chave de C

Categoria	Palavra-chave
Tipos de dados	char, int, float, double, void
Modificadores de tipos	long, short, signed, unsigned
Modificadores de tipos de acesso	const, volatile
Classes de armazenamento	auto, extern, static, register
Tipos definidos pelo programador	struct, enum, union, typedef
Comandos condicionais	if, else, switch, case, default
Comandos de laços	while, for, do
Comandos de desvio	break, goto, return, continue
operador	sizeof

## Programa exemplo 3

- // Programa para calcular a área de um círculo

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

```
int main() {
```

```
    const double PI = 3.14159;
```

```
    double raio = 0.0, area;
```

```
    printf("Digite valor do raio\n");
```

```
    scanf("%f", &raio);
```

```
    area = PI * raio * raio;
```

```
    printf("A area do circulo de raio %f eh %f", raio, area);
```

```
    return 0;
```

```
}
```

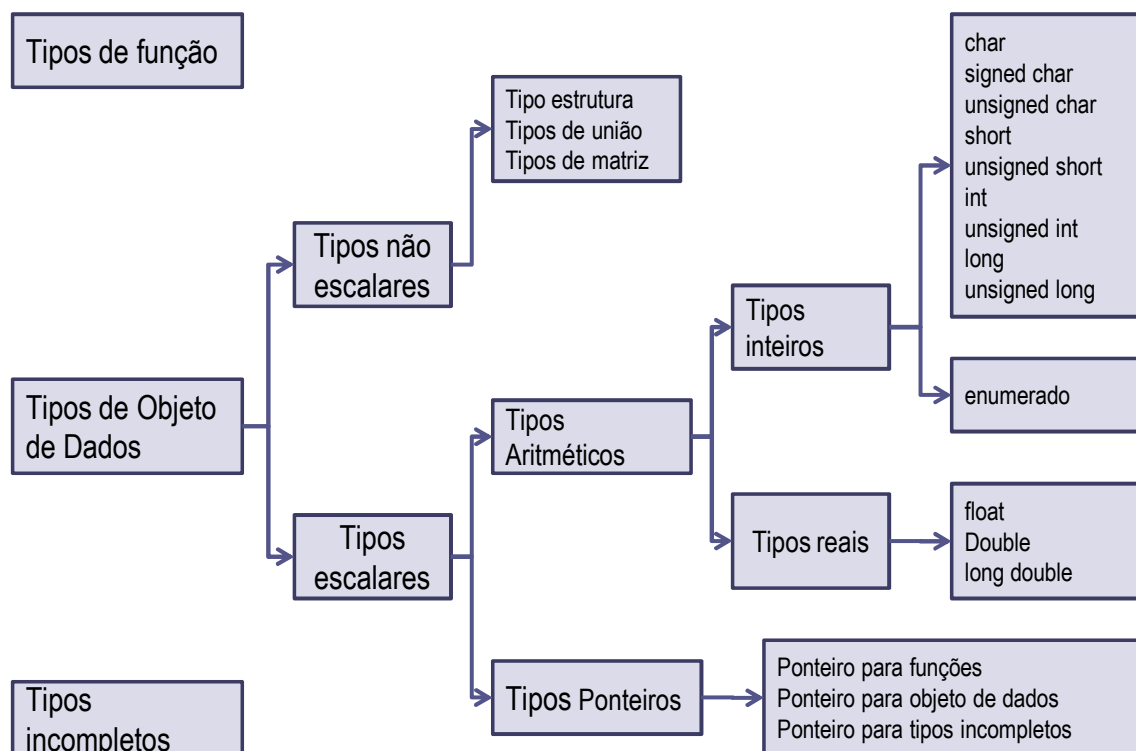
# Estrutura Sequencial

- Comando de **Atribuição**
  - O comando de atribuição é utilizado para atribuir valores ou resultado de expressões a uma variável, sendo representado pelo sinal de igualdade (=)
- forma geral:
  - `variavel = expressao;`
  - A **expressao** pode conter constantes, variáveis, operadores, e chamadas de funções.
  - Ex.:
    - `circ = PI*pow(R,2);`
    - `sexo = 'M';`

# Declaração de Variáveis

- Variáveis devem ser declaradas no início de cada bloco de programa.
- Tipos de Variáveis:
  - **char** (literal/string)
  - **int** (inteiro)
  - **float** (real de precisão simples )
  - **double** (real de precisão dupla)
  - *e outras que serão vistas no momento oportuno*

# Taxonomia de Tipos de Dados



Faculdade da Computação - UFU

21

## Declaração de Variáveis

- A declaração de uma variável consiste em um **tipo** e um **identificador**.
  - O **tipo** determina o espaço de memória que deverá ser alocado.
  - O **identificador** permitirá que ela seja referenciada no restante do programa
- Ex.:

```
tipo var_nome = valor;  
int soma = 0;
```

Faculdade da Computação - UFU

22

# Tipos de Dados

Tipo	Espaço	Escala
char	1 byte	-128 a 127
int	4 bytes	-2.147.483.648 a 2.147.483.647
float	4 bytes	$-3.4 \times 10^{-38}$ a $3.4 \times 10^{+38}$
double	8 bytes	$-1.7 \times 10^{-308}$ a $1.7 \times 10^{+308}$

## Identificadores

- Todo identificador deve iniciar-se com letra (maiúscula ou minúscula) e ser composto exclusivamente por letras, dígitos e sublinhas.
- Faz distinção de letras maiúsculas e minúsculas;
- Não podem ser utilizadas palavras-reservadas
- Ex.:
  - `char tecla_x, opcao;`
    - `int x1, y2, z3;`
    - `float comissao, desconto, salario;`

# Modificadores de Tipo

Short & Long		
Tipo	Espaço	Escala
<b>short int</b>	<b>2 bytes</b>	<b>-2<sup>15</sup> a (2<sup>15</sup>-1)</b>
<b>long int</b>	<b>4 bytes</b>	<b>-2<sup>31</sup> a (2<sup>31</sup>-1)</b>
<b>long double</b>	<b>10 bytes</b>	<b>3.4x10<sup>-4932</sup> a 3.4x10<sup>4932</sup></b>

Signed & Unsigned		
Tipo	Espaço	Escala
<b>unsigned char</b>	<b>1 bytes</b>	<b>0 a 255</b>
<b>unsigned int</b>	<b>4 bytes</b>	<b>0 a 2<sup>32</sup>-1</b>
<b>unsigned long int</b>	<b>4 bytes</b>	<b>0 a 2<sup>32</sup>-1</b>

Segundo o padrão, não existe nenhuma garantia de que uma variável **short int** é menor que uma variável **int**, nem que **long int** é maior que **int**. Apenas é garantido que **int** não é maior que **long** nem menor que **short**. De fato, nos sistemas x86 de 32 bits (ou seja, a maioria dos computadores pessoais atualmente), o tamanho de **int** é igual ao de **long**. Geralmente, **int** será o tamanho nativo do processador — ou seja, **32 bits** num processador de 32 bits, **16 bits** num processador de 16 bits etc.

## Entrada e Saída formatada

- **scanf**

- A função *scanf()* permite que um valor seja lido do teclado e armazenado numa variável.
- Sua sintaxe consiste de uma cadeia de formatação seguida de uma lista de argumentos, indicando o endereço de uma variável:
- `scanf("formatação", arg1, arg2, ..., argn);`

- **Ex.:**

- `int idade;`
- `char sexo;`
- `scanf("%d %c", &idade, &sexo);`



# Entrada e Saída Formatada

Especificadores de Formato	
<b>%c</b>	um único caractere
<b>%o, %d, %x</b>	um número inteiro em octal, decimal ou hexadecimal
<b>%u</b>	um número inteiro em base decimal sem sinal
<b>%ld</b>	um número inteiro longo em base decimal
<b>%f, %lf, %e</b>	um número real de precisão simples ou dupla
<b>%s</b>	uma cadeia de caracteres ( <i>string</i> )
<b>%p</b>	Ponteiro (endereço)
<b>%%</b>	um único sinal de porcentagem

## Entrada e Saída formatada

- **printf**

- A função *printf()* nos permite exibir informações formatadas no dispositivo de saída.
- A sintaxe idêntica àquela da função *scanf()*.
- A principal diferença é que agora a lista de argumentos deve conter os valores a serem exibidos e não seus endereços:

**printf**("*formatação*", **arg1**, **arg2**, ..., **argn**);

- A *cadeia de formatação* pode conter também texto, que é exibido normalmente, e caracteres de controle, cuja exibição causa efeitos especiais



# Entrada e Saída formatada

- Caracteres de Controle

- Os caracteres de controle utilizados com a função ***printf()*** são:

Caractere de controle	Efeito
<b>\a</b>	soa o alarme do microcomputador
<b>\b</b>	o cursor retrocede uma coluna
<b>\f</b>	alimenta página na impressora
<b>\n</b>	o cursor avança para uma nova linha
<b>\r</b>	o cursor retrocede para o início da linha
<b>\t</b>	o cursor avança para próxima marca de tabulação
<b>\"</b>	exibe uma única aspa
<b>\'</b>	exibe um único apóstrofo
<b>\\</b>	exibe uma única barra invertida

## Programa exemplo 4

```
#include <stdio.h>
int main() {
    int n1, n2, n3;
    printf("Entre um número:");
    scanf("%d", &n1);
    printf("Entre outro número:");
    scanf("%d", &n2);
    n3 = n2;
    n3 = n3 - n1;
    printf("a diferença entre %d e %d é: %d", n1, n2, n3);
    return 0;
}
```

# Operadores Matemáticos

Operador	Exemplo	Descrição
+	<b>x + y</b>	Soma o conteúdo de x ao de y
-	<b>x - y</b>	Subtrai o conteúdo de y do conteúdo de x
*	<b>x * y</b>	Multiplica o conteúdo de x pelo conteúdo de y
/	<b>x / y</b>	Obtém o quociente da divisão de x por y
%	<b>x % y</b>	Obtém o resto da divisão de x por y
++	<b>x++</b>	incrementa o conteúdo de x
--	<b>x--</b>	decrementa o conteúdo de x

# Operadores matemáticos de Atribuição

Operador	Exemplo	Descrição
+=	<b>x += y</b>	equivale a $x = x + y$
-=	<b>x -= y</b>	equivale a $x = x - y$
*=	<b>x *= y</b>	equivale a $x = x * y$
/=	<b>x /= y</b>	equivale a $x = x / y$
%=	<b>x %= y</b>	equivale a $x = x \% y$

# Desafio 1

- Diga o que será escrito na tela durante a execução do seguinte trecho de código:


```
int a, b = 0, c = 0;
a = ++b + ++c;
printf("%d %d %d\n", a, b, c);
a = b++ + c++;
printf("%d %d %d\n", a, b, c);
a = ++b + c++;
printf("%d %d %d\n", a, b, c);
a = b-- + --c;
printf("%d %d %d\n", a, b, c);
```

# Programa exemplo 5

```
#include <stdio.h>
int main() {
    int n1, n2, n3;
    printf("Digite um número:");
    scanf("%d", &n1);
    printf("Digite outro número:");
    scanf("%d", &n2);
    n3 = n2++ + ++n1 % 3 * 2 + 7;
    printf("Números: %d, %d e %d", n1, n2, n3);
    return 0;
}
```

# Precedência de Operadores de cima para baixo

## Operadores



```
( [ - .  
! - ++ -- + * & (type-cast) sizeof  
* / %  
+ -  
<< >>  
< <= > >=  
== !=  
&  
^  
|  
&&  
||  
?:  
= += -= *= /= %= &= ^= |= <<= >>=  
,
```

## Associatividade

```
Left to right  
Right to left (+, - and * unários)  
Left to right  
Left to right  
Left to right  
Left to right  
Left to right  
Left to right  
Left to right  
Left to right  
Left to right  
Left to right  
Right to left  
Left to right
```

# Programa exemplo 6

```
#include <stdlib.h>           // diretiva  
/* função principal */  
int main()  
{  
    int i, n, soma;  
    soma=0;  
    printf("N = "); scanf("%d",&n);  
    for(i=0; i<n; i++) { soma += i; }  
    printf("A soma eh: %d\n",soma);  
    return 0;  
}
```

# Bibliografia

- Pereira, S.L., Linguagem C – Distribuição gratuita
- Schildt, H., C Completo e Total, Editora Makron Books do Brasil Editora Ltda, 1996.
- Evaristo, J., Aprendendo a programar programando em linguagem C, Book Express, 2001.
- Mizrahi, V. V., Treinamento em Linguagem C, Curso Completo, Módulos 1 e 2, Makron Books do Brasil Editora Ltda, 1990.
- Kernighan, B.W & Ritchie, D. M., C a Linguagem de Programação, Editora Campus, 1986.

# Dúvidas?



## Desafios:

1. Escreva um programa em C que leia o peso e a altura de uma pessoa e calcule o seu IMC, bem como os valores máximo e mínimo do peso, segundo a OMS, considerados saudáveis.
  - Dado:  $IMC = \text{Peso} / (\text{Altura})^2$ ,  $IMC_{min} = 18,5$ ,  $IMC_{max} = 25$
2. Escreva um programa para calcular o perímetro de uma circunferência de raio  $r$ .
3. Dado a altura de um cilindro circular reto e o raio de sua base, escreva um programa que calcule a área total de sua superfície e o seu volume.