

Programação

Prof. Msc. Silvana Teodoro

Linguagem C

- C é uma **linguagem de programação de computadores**.
- Surgiu nos anos 70
 - Por Dennis Ritchie
 - Lançada com o sistema operacional Unix
- Padrão C ANSI - 1985
 - *American National Standards Institute*
 - Estabelecido padrão para a linguagem
- Portável
 - Compila em qualquer sistema operacional com compilador C ANSI



Linguagem C

- É possível usá-la para criar um conjunto de instruções para que o computador possa executar.
- Ganhou ampla aceitação por oferecer aos programadores o máximo em controle e eficiência.
- C é uma linguagem de sintaxe simples e elegante que permite rápido entendimento pelo programador iniciante.
- Desde sua criação, o C tornou-se uma linguagem popular tanto entre programadores profissionais quanto entre os iniciantes.



Linguagem C

- É a linguagem de programação preferida para o desenvolvimento de sistemas e softwares de base.
- No meio acadêmico, é amplamente utilizada para desenvolvimento de pesquisas científicas e como instrumento de aprendizado para o desenvolvimento de algoritmos.
- Somente para exemplificar o quanto a linguagem C é difundida, podemos citar alguns *softwares* escritos em C: Matlab, Windows, Kernel do Linux, Skype (escrito em C/C++), entre outros.



Linguagem C

- Entre as principais características da linguagem C, podem ser citadas:
 - portabilidade
 - modularidade
 - recursos de baixo nível
 - geração de código eficiente
 - simplicidade
 - facilidade de uso
 - possibilidade de ser usada para os mais variados propósitos
 - indicada para escrever compiladores, editores de textos, bancos de dados, etc.



C e C++

- C++ é uma versão estendida e melhorada de C que foi projetada para suportar programação orientada a objetos.
- C++ contém e suporta toda a linguagem C e mais um conjunto de extensões orientadas a objetos.
- Por muitos anos ainda os programadores escreverão, manterão e utilizarão programas escritos em C.



Compilador GCC

- Todos os programas utilizados como exemplo neste curso utilizam apenas funções da linguagem C padrão ANSI (ANSI C).
- Utilizaremos o compilador GCC do projeto GNU ([http://: www.gnu.org](http://www.gnu.org)), que é gratuito e de código aberto.
- De posse do GCC instalado, para compilar um programa denominado *nome.c*, execute a seguinte linha de comando:
gcc nome.c -o nome.exe
- onde *nome* é o nome dado ao programa. Com isso, o arquivo *nome.c* será compilado gerado um executável *nome.exe* (em ambientes Linux, a extensão *.exe* é desnecessária).

Estrutura básica de um programa C

diretivas para o pré-processador

declaração de variáveis globais

```
main ()
{
    declaração de variáveis locais da função main
    comandos da função main
}
```

```
/* Estrutura de um programa em C */
# include <arquivo_cabecalho.h>
int main ( )
{
    declaração de variáveis
    instrução_1;
    instrução_2;
    função_1(variáveis);
    instrução_3;
    -
    -
}
int função_1 (variáveis)
{
    declaração de variáveis
    instrução_1;
    instrução_2;
    -
    -
    return (INT);
}
```


Estrutura básica de um programa C

- Um programa em C é estruturado sobre funções.
- O programa possui uma ou várias funções, sendo que a principal, que dá início ao programa e chama todas as outras, é sempre chamada **main**.
- Com exceção da **main**, todas as outras funções podem ter qualquer nome que o programador desejar.
- Todo programa C inicia sua execução chamando a função **main()**, sendo obrigatória a sua declaração no programa principal.

Estrutura básica de um programa C

- A geração do programa executável a partir do programa fonte obedece a uma sequência de operações antes de tornar-se um executável:



- Depois de escrever o módulo fonte em um editor de textos, o programador aciona o compilador que no nosso exemplo é chamado pelo comando gcc.
- Essa ação desencadeia uma sequência de etapas, cada qual traduzindo a codificação do usuário para uma forma de linguagem de nível inferior, que termina com o executável criado pelo lincador.

Estrutura básica de um programa C

```
/* programa exemplo 01 */  
# include <stdio.h>  
main ( )  
{  
    printf("Alo, Mundo! ");  
}
```

- Salve este programa com o nome **exemp_01.c** e compile-o utilizando o compilador GCC, ou outro de sua preferência. Escreva na linha de comando do computador:
gcc exemp_01.c -o exemp_01
- Isto indica ao computador para utilizar o compilador GCC para compilar o programa **exemp_01.c** escrito em **C** e dar ao arquivo executável gerado o nome de **exemp_01**.
- Ao final, execute o programa: **.\exemplo_01**
- A função **printf** já está programada dentro de **stdio.h**, e é utilizada para escrever mensagens na tela do computador. As funções pré-programadas devem ser escritas em letras minúsculas.
- Modifique este programa para escrever outra mensagem além de "Alo, Mundo!".

Linguagem C

Variáveis

- Variáveis
 - Declaradas antes de serem usadas
 - Compilador necessita de informações da variável antes de usar
- É um local na memória principal, isto é, um endereço que armazena um conteúdo (informação) que pode ser modificado.
- Em C, não é possível ter variáveis que comecem com dígito e espaços não são permitidos.
 - Exemplos de nomes de variáveis indevidas: 2w, peso do aluno, sal/hora.
- **Observação:** Em C usualmente são utilizadas variáveis em minúsculo e constantes em maiúsculos.

Linguagem C

Variáveis

- Case sensitive

- Funções, variáveis, parâmetros respeitam o que é maiúsculo e minúsculo

```
int dia_atual  ≠  int Dia_atual
```

```
int valorAbsoluto  ≠  int valorabsoluto
```

```
char nome  ≠  char Nome
```

```
ImprimeNF  ≠  ImprimeNf
```

- NÃO é permitido criar variáveis **iguais** às palavras reservadas pelo compilador
- É permitido criar variáveis **semelhantes** às palavras reservadas pelo compilador. **MAS não é aconselhável!**
 - Palavra “while” de laço condicional é diferente de “While”

Linguagem C

Variáveis

- Algumas palavras reservadas pelo compilador
 - 32 palavras
 - Alguns compiladores podem reservar mais palavras

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

Linguagem C

Tipos de Variáveis

- Há cinco tipos básicos de dados em C:
 - caractere (**char**)
 - inteiro (**int**)
 - ponto flutuante(**float**)
 - ponto flutuante de precisão dupla (**double**)

Palavra chave	Tipo	Tamanho	Intervalo
char	Caractere	1	-128 a 127
signed char	Caractere com sinal	1	-128 a 127
unsigned char	Caractere sem sinal	1	0 a 255
Int	Inteiro	2	-32.768 a 32.767
signed int	Inteiro com sinal	2	-32.768 a 32.767
unsigned int	Inteiro sem sinal	2	0 a 65.535
short int	Inteiro curto	2	-32.768 a 32 767
signed short int	Inteiro curto com sinal	2	-32.768 a 32.767
unsigned short int	Inteiro curto sem sinal	2	0 a 65.535
long int	Inteiro long	4	-2.147.483.648 a 2.147.483.647
signed long int	Inteiro longo com sinal	4	-2.147.483.648 a 2.147.483.647
unsigned long int	Inteiro longo sem sinal	4	0 a 4.294.967.295
float	Ponto flutuante com precisão simples	4	3.4 E-38 a 3.4E+38
double	Ponto flutuante com precisão simples	8	1.7 E-308 a 1.7E+308
long double	Ponto flutuante com precisão dupla longo	16	3.4E-4932 a 1.1E+4932

Linguagem C

Declaração e Inicialização de Variáveis

- As variáveis no C devem ser declaradas antes de serem usadas. A forma geral da declaração de variáveis é:

tipo_da_variável lista_de_variáveis;

- As variáveis da lista de variáveis terão todas o mesmo tipo e deverão ser separadas por vírgula.

1. `tipo nome;`

2. `tipo nome = valor;`

1. `int valorA;`

2. `int valorB = 10;`

Linguagem C

Declaração e Inicialização de Variáveis

- Há três lugares nos quais podemos declarar variáveis:
 - O **primeiro** é fora de todas as funções do programa. Estas variáveis são chamadas variáveis globais e podem ser usadas a partir de qualquer lugar no programa.
 - O **segundo** lugar no qual se pode declarar variáveis é no início de um bloco de código de uma função. Estas variáveis são chamadas locais e só têm validade dentro do bloco no qual são declaradas, isto é, só a função à qual ela pertence sabe da existência desta variável.
 - O **terceiro** lugar onde se pode declarar variáveis é na lista de parâmetros de uma função. Mais uma vez, apesar de estas variáveis receberem valores externos, são conhecidas apenas pela função onde são declaradas.

Linguagem C

Declaração e Inicialização de Variáveis

- Podemos inicializar variáveis no momento de sua declaração. Para fazer isto podemos usar a forma geral

tipo_da_variável nome_da_variável = constante;

- Isto é importante pois quando o C cria uma variável ele não a inicializa. Isto significa que até que um primeiro valor seja atribuído à nova variável ela tem um valor indefinido e que não pode ser utilizado para nada.
- Nunca presuma que uma variável declarada vale zero ou qualquer outro valor.

```
char ch='D' ;  
int count=0;  
float pi=3.1416;
```

Linguagem C

Operadores

– Operador de atribuição

- O operador de atribuição em C é o sinal de igual “=”.
- O operador de atribuição “=” atribui à variável à sua esquerda o valor calculado à direita. O C permite que se façam atribuições encadeadas, como:

$$x = y = z = 2.45;$$

- Nesta linha, o valor 2.45 será atribuído a **z**, depois a **y**, depois a **x**.


Linguagem C

Operadores

– Operador Aritméticos

– Usados para desenvolver operações matemáticas.

O Operador de resto (%)
só funciona sobre
inteiros



Aritméticos	Tipo	Operação	Prioridade
+	Binário	Adição	5
-	Binário	Subtração	5
%	Binário	Resto da divisão	4
*	Binário	Multiplicação	3
/	Binário	Divisão	3
++	Unário	Incremento	2
--	Unário	Decremento	2
+	Unário	Manutenção do sinal	1
-	Unário	Inversão do sinal	1

Linguagem C

Operadores

- Operadores aritméticos

Operador	Sintaxe	Exemplo	Valor
Atribuição	<code>x = y</code>	<code>a = 5;</code>	5
Soma	<code>x + y</code>	<code>a = 10+10;</code>	20
Subtração	<code>x - y</code>	<code>a = 10-10;</code>	0
Multiplicação	<code>x * y</code>	<code>a = 5*10;</code>	50
Divisão	<code>x / y</code>	<code>a = 10/2;</code>	5
Módulo (resto)	<code>x % y</code>	<code>a = 23%3;</code>	2
Incremento	<code>x++</code>	<code>a = 0; a++;</code>	1
Decremento	<code>x--</code>	<code>a = 0; a--;</code>	-1

Linguagem C

Operadores

– Operador Aritméticos

- Os operadores de incremento e decremento são unários que alteram a variável sobre a qual estão aplicados.
- O que eles fazem é incrementar ou decrementar de um a variável sobre a qual estão aplicados.
- Então:

`x++;`

`x--;`

- são operações equivalentes a

`x = x+1;`

`x = x-1;`

Linguagem C

Operadores

– Operador Aritméticos

- Estes operadores podem **ser pré-fixados** ou **pós-fixados**.
- A diferença é que quando são pré-fixados eles incrementam e retornam o valor da variável já incrementada. Quando são pós-fixados eles retornam o valor da variável sem o incremento e depois incrementam a variável.

- Então, em

```
x = 23;  
y = x++;
```

- teremos, no final, y = 23 e x = 24.

```
x = 23;  
y = ++x;
```

- teremos, no final, y = 24 e x = 24.

Linguagem C

Operadores

- Operador Aritméticos

- Quando efetuados sobre um mesmo operando, podem ser utilizados em forma reduzida

<code>i += 2;</code>	\longrightarrow	<code>i = i+2;</code>
<code>x *= y+1;</code>	\longrightarrow	<code>x = x*(y+1);</code>
<code>d -= 3;</code>	\longrightarrow	<code>d = d - 3;</code>

Linguagem C

Operadores

- Operadores aritméticos

Operador	Sintaxe	Equivalência	Exemplo	Valor
Atribuição por soma	<code>x += y</code>	<code>x = x + y;</code>	<code>x = 1;</code> <code>x += 2;</code>	3
Atribuição por subtração	<code>x -= y</code>	<code>x = x - y;</code>	<code>x = 1;</code> <code>x -= 2;</code>	-1
Atribuição por multiplicação	<code>x *= y</code>	<code>x = x * y;</code>	<code>x = 2;</code> <code>x *= 2;</code>	4
Atribuição por divisão	<code>x /= y</code>	<code>x = x / y;</code>	<code>x = 2;</code> <code>x /= 2;</code>	1
Atribuição por módulo	<code>x %= y</code>	<code>x = x % y;</code>	<code>x = 5;</code> <code>x %= 2;</code>	1

Linguagem C

Operadores

– Operadores relacionais e lógicos

Relacionais		Lógicos	
>	Maior que	&&	And (e)
>=	Maior ou igual		Or (ou)
<	Menor que	!	Not (não)
<=	Menor ou igual		
==	Igual		
!=	Diferente		

Linguagem C

Operadores

- Operadores de comparação (condicionais)

Operador	Sintaxe
Igualdade	<code>x == y</code>
Diferença	<code>x != y</code>
Maior que	<code>x > y</code>
Maior ou igual	<code>x >= y</code>
Menor que	<code>x < y</code>
Menor ou igual	<code>x <= y</code>

CUIDAR

```
1. int x = 0;  
2. // Condicional se x é igual a 0  
3. if (x == 0) { ... }
```

DIFERENTE

```
1. int x = 0;  
2. // atribui 0 ao x, sem erro de sintaxe  
3. if (x = 0) { ... }
```

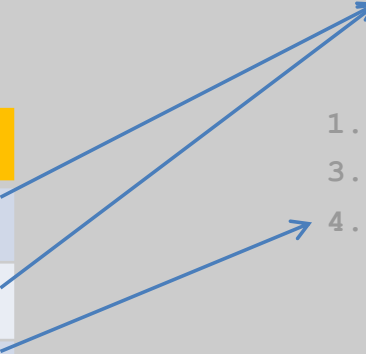
Linguagem C

Operadores

- Operadores de lógicos

Operador	Sintaxe
NOT	!x
AND	x && y
OR	x y

```
1. int x = 5;  
3. // Se x NÃO estiver entre 0 e 10  
4. if (!(x > 0 && x < 10)) { ... } // FALSE
```



```
1. int x = 5;  
3. // Se x for divisível por 5 ou maior que 100  
4. if ((x%5) == 0 || x > 100) { ... } // TRUE
```

Linguagem C

Operadores

– Operador **sizeof**

- O operador **sizeof** é usado para se saber o tamanho de variáveis ou de tipos. Ele retorna o tamanho do tipo ou variável em **bytes**.
- O **sizeof** admite duas formas:

sizeof nome_da_variável

sizeof (nome_do_tipo)

- Se quisermos saber o tamanho de um **float** fazemos **sizeof(float)**. Se declararmos a variável **f** como **float** e quisermos saber o seu tamanho faremos **sizeof(f)**.

Linguagem C

Diretivas para o processador - Bibliotecas

- Diretiva `#include` permite incluir uma biblioteca
- Bibliotecas contêm funções pré-definidas, utilizadas nos programas
- Exemplos

<code>#include <stdio.h></code>	Funções de entrada e saída
<code>#include <stdlib.h></code>	Funções padrão
<code>#include <math.h></code>	Funções matemáticas
<code>#include <string.h></code>	Funções de texto

Linguagem C

Funções Matemáticas

Função	Exemplo	Comentário
<code>ceil</code>	<code>ceil(x)</code>	Arredonda o número real para cima; <code>ceil(3.2)</code> é 4
<code>cos</code>	<code>cos(x)</code>	Cosseno de x (x em radianos)
<code>exp</code>	<code>exp(x)</code>	e elevado à potencia x
<code>fabs</code>	<code>fabs(x)</code>	Valor absoluto de x
<code>floor</code>	<code>floor(x)</code>	Arredonda o número real para baixo; <code>floor(3.2)</code> é 3
<code>log</code>	<code>log(x)</code>	Logaritmo natural de x
<code>log10</code>	<code>log10(x)</code>	Logaritmo decimal de x
<code>pow</code>	<code>pow(x, y)</code>	Calcula x elevado à potência y
<code>sin</code>	<code>sin(x)</code>	Seno de x
<code>sqrt</code>	<code>sqrt(x)</code>	Raiz quadrada de x
<code>tan</code>	<code>tan(x)</code>	Tangente de x

```
#include <math.h>
```

Linguagem C

Comentários

- Simples

```
// Este é um comentário simples
```

- Em várias linhas

```
/* Este é um  
   comentário  
   em várias  
   linhas  
*/
```

```
// Função para impressão de NF  
int ImprimeNF(int ID_NotaFiscal)  
{  
  
    /*  
     * Localiza a NF e imprime  
     */  
  
    ...  
  
}
```


Linguagem C

Comandos de Entrada e Saída

- Função **printf()**
 - Imprime conteúdo em tela
 - Pode imprimir mensagens e variáveis
 - Bastante usado para depuração de programas

Sintaxe

1. **printf**(string_controle, lista_argumentos);

- **string_controle**
 - É o que a função imprimirá na tela
 - Pode-se adicionar parâmetros na mensagem
- **lista_argumentos**
 - São os valores dos parâmetros da mensagem

Linguagem C

Comandos de Entrada e Saída

- Função **printf()**

printf()	
Código	Tipo
%d	Inteiro
%f	Float
%c	Caractere
%s	String
%%	Imprime %

```
#include <stdio.h>
main( )
{
    printf("\n%4.2f", 3456.78);
    printf("\n%3.2f", 3456.78);
    printf("\n%3.1f", 3456.78);
    printf("\n%10.3f", 3456.78);
}
```

Linguagem C

Comandos de Entrada e Saída

- Função **printf()**

Exemplos

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("%s está a %d milhões de milhas do sol","Vênus",67);
5. }
```

```
1. #include <stdio.h>
2. int main( )
3. {
4.     printf("Valor inteiro atribuído foi %d para o caracter %c e um float
5.         foi de %f ",99,a,1.45);
6. }
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("Se quisesse imprimir uma string : %s","Minha string!");
5. }
```

Linguagem C

Comandos de Entrada e Saída

- Função **printf()**

Exemplo

```
1. void imprime_soma(int x, int y) {
2.     printf("Soma: %d", (x+y));
3. }
4.
5. void imprime_pi() {
6.     float fPi = 3.1415;
7.     double dPi = 3.1415;
8.     printf("Float Pi: %f || Double Pi: %f", fPi, dPi);
9. }
10.
11. void imprime_string() {
12.     char string[50] = "Linguagem de Programação I";
13.     printf("Segundo caractere da string %s é %c", string, string[1]);
14. }
```

Linguagem C

Comandos de Entrada e Saída

–Impressão de Códigos Especiais

Código	Ação
<code>\n</code>	Leva o cursor para a próxima linha
<code>\t</code>	Executa uma tabulação
<code>\b</code>	Executa um retrocesso
<code>\f</code>	Leva o cursor para a próxima página
<code>\a</code>	Emite um sinal sonoro (<i>beep</i>)
<code>\"</code>	Exibe o caractere <code>"</code>
<code>\\</code>	Exibe o caractere <code>\</code>
<code>% %</code>	Exibe o caractere <code>%</code>

```
int main( )  
{  
    Printf("\n");  
}
```

Linguagem C

Comandos de Entrada e Saída

–Impressão de Números com Casas Decimais

```
#include <stdio.h>

int main()
{
    printf("Default: %f \n",3.1415169265);
    printf("Uma casa: %.1f \n",3.1415169265);
    printf("Duas casas: %.2f \n",3.1415169265);
    printf("Três casas: %.3f \n",3.1415169265);
    printf("Notação Científica: %e \n",3.1415169265);
}
```

Linguagem C

Comandos de Entrada e Saída

- Função **scanf ()**
 - Ler dados de entrada do teclado
 - Valor digitado é armazenado em uma variável pré-definida

Sintaxe

1. **scanf**(string_controle, &lista_argumentos) ;

- string_controle
 - É o tipo de dado lido do teclado
- lista_argumentos
 - São os valores dos parâmetros da mensagem
 - **&** utilizado para obter o endereço de memória da variável.
 - Devem ser separados por vírgulas

scanf()	
Código	Tipo
%d	Inteiro
%f	Float
%c	Caractere
%s	String
%%	Imprime %

Linguagem C

Comandos de Entrada e Saída

- Função **scanf()**

Exemplo

```
1. int main ()
2. {
3.     int idade;
4.     printf("Digite a sua idade");
5.     scanf("%d",&idade);
6.     printf("A sua idade é %d",idade);
7.     return 0;
8. }
```


Linguagem C

Comandos de Entrada e Saída

- Função **scanf()**

Exemplo

```
1. void imprime_soma() {  
2.     int valor_A, valor_B, soma;  
3.     printf("Digite um número: ");  
4.     scanf("%d", &valor_A);  
5.  
6.     printf("Digite outro número: ");  
7.     scanf("%d", &valor_B);  
8.  
9.     soma = valor_A + valor_B;  
10.    printf("Soma: %d", soma);  
11. }
```

Linguagem C

- Exemplo de um programa C

```
1. #include <biblioteca.h> // Diretivas de pré-processamento
2.
3. int valorTotal;          // Declaração de variáveis globais
4.
5. int multiplica(int,int); // Declaração de protótipo de funções
6.
7. void main()              // Programa principal
8. {
9.     int valorA, valorB;   // Declaração de variáveis locais
10.
11.     valorA = 10;
12.     valorB = 8;
13.     valorTotal = multiplica(valorA, valorB); // Chamada de função
14. }
15.
16. int multiplica(int vA, int vB) // Definição de operações de função
17. {
18.     return vA*vB; // Valor de retorno da função, se não for do tipo void
19. }
```

Linguagem C

Dicas

- Termine todos os comandos com ;
- Quando ocorrer um erro de compilação, verifique a linha do erro e também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o ;

EXERCÍCIOS



Linguagem C

- 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.
- 3) Faça um programa que imprima o numero 75.7632489 com 1,2 e 5 casas decimais.

Linguagem C

4) Ler do teclado dois números inteiros

– Desenvolver 4 operações

- **Somar** os dois números e imprimir o resultado em tela
- **Subtrair** os dois números e imprimir o resultado em tela
- **Dividir** os dois números e imprimir o resultado em tela
- **Multiplicar** os dois números e imprimir o resultado em tela

Linguagem C

- 5) Faça um programa que peça ao usuário para digitar sua altura, leia o dado digitado e imprima o valor.
- 6) Faça um programa o qual o usuário digite seu peso e o de mais duas pessoas, imprima os valores digitados.