

UNILINS – Centro Universitário de Lins

---

**Disciplina : Programação Estruturada**



# A LINGUAGEM C

---

*Prof. Esp. Adalberto Sanches Munaro*  
munaro@unilins.edu.br

# A Linguagem C

---

Estaremos estudando aqui a forma de programar no “**ANSI-C standard**”. A linguagem C foi introduzida pelos pesquisadores Kernigan e Ritchie através do livro “**The C Programming Language**”. Atualmente já existe a segunda versão deste livro: “**The C Programming Language - Second Edition**” que trata da forma “clássica” de programação em C.

Todos os programas em ANSI C possuem um bloco principal de programação, uma função chamada de **main()**.

# A Linguagem C

---

**As funções** são as entidades operacionais básicas dos programas em C, que por sua vez são **uma união de uma ou várias funções executando cada qual o seu trabalho.**

Há funções básicas que estão definidas em **bibliotecas em C** (**arquivos .h**).

As funções **printf()** e **scanf()** por exemplo, permitem **respectivamente** escrever informações na tela e ler dados à partir do teclado – estas funções estão definidas na biblioteca de nome **<stdio.h>**.

Todo programa em ANSI C inicia sua execução chamando a função **main()**, **sendo obrigatória a sua** declaração do dentro do programa.

# A Linguagem C

---

A função **main( )** é especial por ser a primeira a ser chamada quando seu programa é executado.

Significa o **início de seu programa.**

É obrigatória a existência **de main( )** em **algum ponto do programa**, para que o compilador C consiga determinar onde iniciar a **execução.**

# Estrutura de um Programa em C

---

```
/*  definições de pré-processamento  */  
    . . .  
/*  declarações de variáveis globais  */  
    . . .  
main( )  
    {  
        /*  declarações de variáveis locais  */  
        . . .  
        /*  corpo do programa principal  */  
        . . .  
    }  
/*  definição de outras funções  */
```

# Código em C/C++

---

```
/* Este programa mostra mensagens e dados na tela */  
  
#include <stdio.h>  
#include <stdlib.h>  
#define ano 2023  
main( )  
{ char MES [10]="Agosto";  
  printf(" ** FPTE - UNILINS ** ");  
  printf("\n\n ** %s de %d ** ",MES,ano);  
  system("pause > null");  
}
```

A declaração **#include** serve para referenciar arquivos de bibliotecas.

No exemplo anterior a biblioteca referenciada é a **<stdio.h>**, a qual é a biblioteca padrão para as funções de I/O (Entrada/Saída) e que contém a função **printf( )** utilizada no corpo da função **main( )**

Os blocos são delimitados por abre-chave  
"{" para indicar o início do bloco e fecha-chave  
"}" para indicar o fim do bloco

Os comentários na linguagem C devem estar entre /\* ..... \*/

**A linguagem C é sensível ao case**, ou seja, a linguagem diferencia letras **maiúsculas** de letras **minúsculas**. Assim, todo comando de C deve ser escrito **em letras minúsculas**

# Identificadores

---

Todo **Léxico** que não pertencer a gramática de uma linguagem de programação deverá ser **identificado e definido**.

Os **identificadores** nomeiam as **variáveis**, as **constantes**, **tipos das variáveis**, **funções** e até mesmo o **controle do fluxo** de um programa escrito em C.

- o conjunto de caracteres que podem compor um identificador são letras maiúsculas e minúsculas, sublinha( **\_** ) e dígitos;
- **todo** identificador começa com **letra** ou **sublinha ( **\_** )**;

**a) idade**

**b) Nome\_pessoa**

**c) Base**

**d) Altura\_1**

**e) abc\_123**

**f) \_a1\_b2\_c3**



# Definição de Variáveis em C

---

Um computador necessita organizar os dados que manipula na forma de **variáveis**.

**Como o próprio nome** indica, seu conteúdo é “**variável**”, isto é, serve para armazenar valores que podem ser modificados durante a execução de um programa.

**<tipo> lista\_de\_identificadores\_de\_variáveis;**

**Exemplos:**

**int idade**

**float x, y;**

**char opcao;**

**char nome[12]**

# Declaração e Inicialização de Variáveis

---

Para declarar uma variável de ***nome idade do tipo inteiro*** em um programa C, a seguinte expressão seria utilizada:

**int idade;**

Essa declaração **reserva na memória um espaço para a variável i**, suficiente para armazenar a representação **binária em complemento de dois (com sinal) do valor associado à variável**, que inicialmente é indefinido.

É possível, ao mesmo tempo em que se declara uma variável, proceder a sua **inicialização, isto é, se atribuir** um valor inicial definido para a variável, neste caso, a declaração ficaria assim:

**int idade = 0;**

# Tipos de Dados Básicos

---

Como podemos perceber pelo item anterior, o computador exige que o usuário defina **uma variável relacionada cada dado à ser manipulado.**

Tipo de Dado	Bytes Exigidos	Faixa de Variação
<i>char</i>	1	0 à 255
<i>int</i>	2	-32.768 à 32.767
<i>float</i>	4	$3.4 \times 10^{\pm 38}$ (7 dígitos)
<i>double</i>	8	$1.7 \times 10^{\pm 308}$ (15 dígitos)

# Declaração, Inicialização e atribuição de Variáveis

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
    int cont=125;
    float taxa=1.2, taxa_anual=1.4;
    char sexo ='F';
    char nome [10]="Raquel"
    printf("\n Nome   =   %s ", nome);
    printf("\n Sexo    =   %c ", opcao);
    printf("\n Conta   =   %d ", cont);
    printf("\n Valor de taxa   = %f", taxa);
    printf("\n Valor de taxa_anual = %f", taxa_anual);
    system ("pause > null");
} /*Fim função main*/
```

# Formatação de Casas Decimais

É possível também indicar o tamanho do campo, justificar o números de casas decimais. Para isto usa-se códigos colocados entre o **%** e a letra que indica o **tipo de formato**.

Comando - `printf (" %02d + %5.2f = %6.4f ", 3, 4.567, 3 + 4.567)`

Saída - `03 + 4.57 = 7.5670`

Preenche com  
zeros as esquerda  
a quantidade de  
casas faltante

5 caracteres  
e  
2 casas decimais

6 caracteres  
e  
4 casas decimais

# Operadores Aritméticos ( + - \* / % (módulo))

---

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
    int a,b,c,d;
    a = 4;
    b = a*2;
    c = b /1;
    d = c%a;
    printf("\n a = %d  b = %d  c = %d  d = %d",a,b,c,d);
} /*Fim main*/
```

**Valores de a=4   b= 8   c = 8   d=0**

# Formatos de saída de dados na tela

---

**%s** para imprimir string

**%c** para caractere

**%d** para decimal

**%i** para int

**%f** para float

**%x** para hexadecimal

**%e** para notação científica

**%o** para octal

**%u** para decimal sem sinal

**%ld** para decimal longo

**%lf** para ponto flutuante longo (double)

## Comando para Saída de Dados -> Função : printf ()

---

**printf ("Mensagem % caracteres\_controle", dados);**

A função printf() pode imprimir **mensagens e dados simples na tela** (Monitor).

```
#include <stdio.h>
main( )
{
    int x=3;
    printf ("\n Qual é o valor de X ");
    printf ("\n X vale = %d ",x);
    printf ("\n teste..");
}
```



# Caracteres Especiais

---

**\a** apito

**\b** retrocede uma posição

**\\** o caractere \

**%%** o caracter %

**\n** nova linha e outros

**\t** coloca um TAB

**\”** o caractere aspas (“)

**\f** salta página de formulário

## Comando para Entrada de Dados -> Função scanf()

---

A função **scanf()** é para **entrada de dados** à partir do **teclado**, implementada em todos os compiladores C (por padrão).

Ela seria o complemento para a função **printf()**

# Comando para Entrada de Dados -> Função scanf()

---

**scanf( “expressão de controle”, lista de argumentos )**

onde “*expressão\_de\_controle*” serve para indicar o tipo de dado à ser lido : %d,%f,%s e %c

e “*argumentos*” contém o **nome da variável** onde o dado lido através do teclado **vai ser depositado**

**scanf ("%d", & idade);**

O símbolo & é utilizado para expressar o endereço de memória da variável que vai armazenar o valor de entrada.

Quando o valor de entrada é uma **string** (vetor de caracteres) não é necessário colocar o &, pois a linguagem C já trata uma string pelo endereço de seu primeiro caractere.

## Função scanf() - Exemplo

---

```
#include<stdio.h>
main ()
{
    int a, b, Soma = 0;
    printf ("\n Entre com a: ");
    scanf ("%d", & a);
    printf ("\n Entre com b: ");
    scanf ("%d", & b);
    Soma = a + b;
    printf ("\n O valor de c e' = %d ", Soma);
}
```

# Função getchar()

---

A função **getchar()** lê uma *caracter inserido pelo teclado* e a devolve para a variável um caracter passado como argumento da função

Esta função é **nativa do Compilador C/C++**

```
# include <stdio.h>
main( )
{
    char sexo;
    printf (“ \n Digite o sexo F/M :“);
    fflush(stdin);
    sexo = getchar( );
    printf (“ \n O seu sexo = %c \n “, sexo);
}
```

# Função gets()

---

A função **gets()** lê uma *string de caracteres inserida pelo teclado* e a devolve para a variável string passada como argumento da função.

Esta função é **nativa do Compilador C/C++**

```
# include <stdio.h>
main ( )
{
    char nome [30];
    printf (“\n Digite o seu nome: “);
    fflush(stdin);
    gets (nome);
    printf (“\n Oi ... %s !“, nome);
}
```

# Função **getche()**

---

Lê um caractere da entrada padrão. Com a função **getche( )** não é necessário digitar **<ENTER>** após digitar o caractere.

Esta função está definida na biblioteca “**conio.h**”

```
#include <stdio.h>
#include <conio.h>
main( ) {
    char ch;
    printf("\n Digite algum caractere: ");
    fflush(stdin);
    ch=getche( );
    printf("\n A tecla digitada foi %c.", ch);
}
```

# Função getch()

---

Lê um caractere da entrada padrão. Com a função **getch( )** não é necessário digitar **<ENTER>** após digitar o caractere e o caractere digitado não aparecer na tela.

Esta função está definida na biblioteca “**conio.h**”

```
#include <stdio.h>
#include <conio.h>
main( ) {
    char ch;
    printf("\n Digite algum caractere: ");
    fflush(stdin);
    ch=getch( );
    printf("\n A tecla digitada foi %c.", ch);
}
```



# O que é uma cadeia de caracteres (ou string)?

---

Uma **cadeia de caracteres** (string em inglês) é uma sequência de caracteres, ou seja, um conjunto de símbolos, que fazem parte do conjunto de caracteres, definido pelo código ASCII.

Em linguagem C, uma cadeia de caracteres é uma tabela, com vários dados do tipo char, cujo último elemento é o caractere nulo '\0', ou seja, o primeiro caractere do código ASCII (cujo valor é 0).

## A biblioteca string

A **biblioteca string** da linguagem C contém várias funções de manipulação de strings. Para usar essas funções, o seu programa deve incluir a interface string.h:

**#include <string.h>**

## COPIANDO STRINGS: STRCPY()

---

### **strcpy (string\_destino, string\_origem);**

Realiza a cópia do conteúdo de uma variável a outra.

Obs : Ambas devem ser strings.

```
#include <stdio.h>
#include <string.h> //necessário para strcpy
#include <conio.h>
int main (void)
{
    char nome[15];

    strcpy(nome, "Fulano de Tal");
    //strcpy(string_destino, string_origem);
    //note que a string de destino é nome
    //a string de origem é "Fulano de Tal"

    printf("Nome = %s", nome);

    getch();
    return 0;
}
```

# COMPARANDO STRINGS: STRCMP()

---

**strcmp (string1, string2);**

**Compara o conteúdo de duas strings;**

**Possíveis valores de retorno:**

0: conteúdo das strings são iguais, < 0: conteúdo da string1 é menor do que string2, > 0: conteúdo da string1 é maior do que string2

```
#include <stdio.h>
#include <string.h>
main()
{
    char senhaCadastrada[6]="12345";
    char senhaDigitada[6];
    printf("\n Digite Uma Senha:");
    gets(senhaDigitada);
    if(strcmp(senhaCadastrada,senhaDigitada)==0)
        printf("\n Senha Digitada esta Correta.. ");
    else
        printf("\n Senha Digitada esta Incorreta.. ");
}
```