

# **Algoritmos e Programação II**

## **Introdução a Linguagem C**

Profa. Dra. Eloize Seno





# Visão Geral da Linguagem C

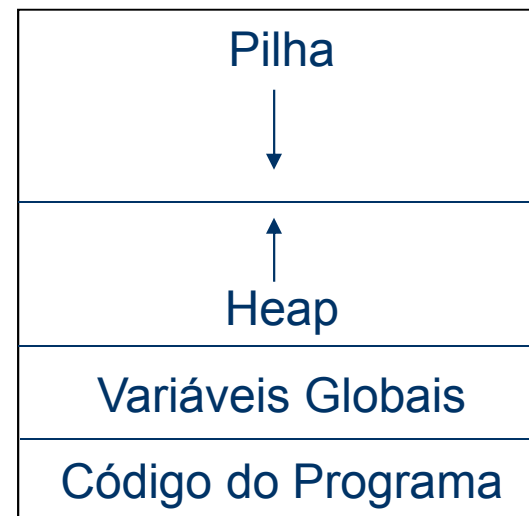
- Desenvolvida por Dennis Ritchie, originada da linguagem B, proposta por Ken Thompson sobre forte influência da linguagem BCPL
- Características:
  - Estruturada
  - Nível médio: permite a manipulação de bits, bytes e endereços
  - Facilmente portátil
  - Compilada

## Visão Geral da Linguagem C (cont.)

- Compilação *versus* Interpretação
  - Interpretador: lê o código-fonte uma linha por vez e executa cada instrução específica;
  - Compilador: lê o código-fonte todo e converte-o em um código-objeto (tradução para linguagem de computador);

## Visão Geral da Linguagem C (cont.)

- Biblioteca C:
  - Conjunto de funções que realizam as tarefas mais comuns
  - Possui linkeditor (linker)
- Mapa de memória:





## Visão Geral da Linguagem C (cont.)

- Forma geral de um programa em C

```
#include <nome_da_biblioteca>
```

```
declarações globais
```

```
tipo de retorno main()  
{  bloco de comandos }
```

```
tipo de retorno f1(lista de parâmetros)  
{  bloco de comandos }
```

```
...
```

```
tipo de retorno fn(lista de parâmetros)  
{  bloco de comandos }
```

# Tipos de Dados

- O tipo de uma variável define os valores que ela pode assumir e as operações que podem ser realizadas com ela
- Tipos básicos: **char**, **int**, **float**, **double** e **void**

**Obs:** Em C não existe o tipo de dado ***boolean***. Qualquer valor diferente de 0 é tratado como verdadeiro.

## Tipos de Dados (cont.)

- **char**: um byte que armazena o código de um caractere do conjunto de caracteres local
- **int**: um inteiro cujo tamanho depende do processador e do compilador usado, tipicamente 16 ou 32 bits (2 ou 4 bytes)
- **float**: um número real com precisão simples
- **double**: um número real com precisão dupla

# Modificadores de Tipos

- Os modificadores alteram algumas características dos tipos básicos para adequá-los às necessidades específicas
- Modificadores:
  - **signed**: indica número com sinal (inteiros e caracteres)
  - **unsigned**: número apenas positivo (inteiros e caracteres)
  - **long**: aumenta a precisão (inteiros e reais)
  - **short**: reduz a precisão (inteiros e reais)





# Identificadores

- São nomes de variáveis, funções, etc.
- Regra de formação:
  - Primeiro caractere: deve ser uma letra ou um sublinhado
  - Caracteres subsequentes: podem ser letras, números ou sublinhados
  - Exemplos: `cont`, `test10`, `tam_largura`

**Atenção:** Letras maiúsculas e minúsculas são tratadas diferentemente. Logo, `cont`, `Cont` e `CONT` são 3 identificadores distintos.



# Variáveis

- Forma geral:  
    tipo lista\_de\_variáveis;
- Exemplos:  
    int i, j, l;  
    unsigned int y;  
    char x;  
    double saldo, lucro, prejuizo;
- Declarações: dentro de funções, na definição dos parâmetros das funções, fora de todas as funções.



# Operadores Aritméticos

- Subtração –
- Adição +
- Multiplicação \*
- Divisão /
- Módulo da divisão (resto) %
- Decremento – –
- Incremento ++



# Operadores Relacionais

- Comparação entre valores:
  - Maior >
  - Menor <
  - Maior ou igual >=
  - Menor ou igual <=
  - Igual ==
  - Diferente !=



# Operadores Relacionais (cont.)

- Precedência (Esq  $\rightarrow$  Dir)
  - $>$ ,  $<$ ,  $>=$ ,  $<=$
  - $!=$ ,  $==$
- Retorna sempre 0 (falso) ou 1 (verdade)
  - $j = 7$
  - $3 < j < 5$



# Operadores Lógicos

- Negação → !
- E (and) → &&
- Ou (or) → ||
- Precedência  
! → && → ||



# Precedência dos Operadores

- Parentêses  $\rightarrow$  ( )
- !
- Operadores aritméticos
- Operadores relacionais
- Operadores lógicos



# Comandos de Atribuição

- Representado pelo sinal de igualdade (=)

- Exemplos:

- $x = 4;$

- $x = x + 2;$

- $y = 2.5;$

- $\text{sexo} = \text{'F'};$

**Cuidado:** Não confundir Atribuição (=)  
com Igualdade (==)

- Múltiplas atribuições:

- C permite a atribuição de mais de uma variável em um mesmo comando:

- Ex:  $x = y = z = 0;$





# Comentários

- Em bloco: `/* */`
  - Ex: `int i, j, l; /* declaração de variáveis em C */`
- Linha única: `//`
  - Ex: `int i, j, l; // declaração de variáveis`

# Comandos de Entrada

- Utilizado para receber dados fornecidos pelo usuário (dados de entrada) e armazená-los na memória principal (em variáveis)
- Os dados são fornecidos ao sistema por meio de um dispositivo de entrada, cuja configuração dada como padrão é o teclado.
- A linguagem C oferece vários comandos de entrada, cada qual mais indicado para uma situação em particular.
- O comando de entrada mais comum é o **scanf**

## Comandos de Entrada (cont.)

- Sintaxe: **scanf**("formato", &variável);
- Funcionamento:
  - O comando coleta as informações dadas no dispositivo padrão de entrada, interpretando as informações segundo a máscara de formatação e armazenando na(s) respectiva(s) variável(is) dada(s).
  - O dispositivo padrão é dado pela variável **stdin**

## Comandos de Entrada (cont.)

- Entrada formatada **scanf()**.  
Exemplos: **int idade; float salario; double x;**  
`scanf("%d",&idade);`  
`scanf("%f",&salario);`  
`scanf("%lf",&x);`
- Ou ainda: **int dia, mes, ano;**  
`scanf("%d%d%d", &dia, &mes, &ano);`
- **OBS:** o comando **scanf** armazena toda a cadeia de caracteres até que seja pressionado "enter" ou até que encontre um caractere em branco



# Máscara de Formatação

- Símbolo ‘%’ seguido de uma letra:
  - %c Caractere
  - %d Inteiros com sinal
  - %s Cadeia de caracteres (strings)
  - %u Inteiros sem sinal
  - %f Números reais (float)
  - %lf Números reais (double ou long float)
  - %e Notação científica
  - %x Números em hexadecimal

## Comandos de Entrada (cont.)

- **gets**: armazena cadeias de caracteres (inclusive espaços em branco, como um nome completo)

Exemplo:

`gets(Nome);`      /\* um ou mais caracteres são armazenados em Nome \*/

- Os comandos **gets** e **scanf** exigem a inclusão da biblioteca **stdio.h**

# Comandos de Saída

- Empregados para que o sistema forneça, em um dispositivo de saída, as mensagens e resultados de seu processamento.
- O dispositivo padrão de saída é o monitor.
- A linguagem C oferece alguns comandos de saída, mas o que apresenta propósito mais geral é o **printf**.



# Comandos de Saída

- Sintaxe:
  - `printf("Mensagem", lista de variáveis);`
- Ex: `printf("Hello world!");`
- O comando **printf** exige a inclusão da biblioteca **stdio.h**



## Comandos de Saída (cont.)

- Saída formatada **printf()**.
- Exemplo:
  - **int i = 10;**
  - **float r = 3.1514;**
  - **char s = 'A';**
  - **printf("Inteiro: %d, Real: %f, Caractere: %c",i,r,s);**
- Produz:  
Inteiro: 10, Real: 3.151400, Caractere: A



# Constantes do tipo char

- Barra invertida seguido de um caractere:
  - \a bip
  - \b backspace
  - \n nova linha
  - \t tabulação horizontal
  - \' apóstrofe
  - \" aspas
  - \\ barra invertida

# Principais Funções Matemáticas

- A Linguagem C disponibiliza, por meio da biblioteca **math.h**, diversas funções para cálculos. Alguns exemplos de funções:
  - **ceil(X)**: arredonda um número real para cima. Ex:  $\text{ceil}(3.2) = 4$
  - **floor(X)**: arredonda um número real para baixo. Ex:  $\text{floor}(3.2) = 3$
  - **sqrt(X)**: calcula a raiz quadrada de X
  - **cbrt(X)**: calcula a raiz cúbica de X
  - **cos(X)**: retorna o co-seno de X (X deve estar em radianos)
  - **sin(X)**: retorna o seno de X (X deve estar em radianos)
  - **tan(X)**: retorna a tangente de X (X deve estar em radianos)

# Principais funções matemáticas - (cont.)

- **log(X)**: retorna o logaritmo natural de X
- **log10(X)**: retorna o logaritmo de base 10 de X
- **pow(X,Y)**: retorna a potência de X elevada a Y
- **abs(X)**: retorna o valor absoluto de X
- **M\_PI**: retorna o valor de  $\pi$
- **modf** (Ex. de uso:  $z = \text{modf}(X, \&Y)$ ): decompõe o número real armazenado em X em duas partes. Y recebe a parte inteira e z, a parte fracionária do número.



# Fluxo de execução de um programa

- Estruturas básicas de controle do fluxo de execução:
  - Estrutura sequencial
  - Estrutura condicional (ou de seleção)
  - Estrutura de repetição

# Estrutura Sequencial

- Estrutura sequencial: o conjunto de ações é executado em sequência linear de cima para baixo e da esquerda para a direita.
- **Importante:** a execução inicia sempre pela função `main()`



# Estrutura Condicional ou de Decisão (Desvio/Seleção)

- Permite a escolha de um grupo de ações (bloco) a ser executado, quando determinadas condições forem ou não satisfeitas.
- As estruturas condicionais podem ser classificadas em: simples, compostas e aninhadas.
  - Comandos: **if-else**, **case**

# Estrutura Condicional Simples

- Forma 1: comando único

```
if (condição)  
    comando;
```

- Onde: <condição> é uma expressão lógica que pode gerar um resultado *verdadeiro* ou *falso*

Exemplo:

```
if (n1 >= 0)  
    printf("Número positivo!");
```





# Estrutura Condicional Simples

- Forma 2: bloco de comandos

```
if (condição)
{
    comando1;
    comando2;
    comando3;
}
```

- **Atenção:** o uso de chaves é **obrigatório** somente quando há mais de um comando.  
DICA: usar sempre que estiver em dúvida!

# Estrutura Condicional Composta

- É usada em situações em que duas ou mais alternativas dependem de uma mesma condição: uma depende da condição ser *verdadeira* e a outra da condição ser *falsa*.

- Forma 1: comando único

```
if (condição)
    comando1;
else
    comando2;
```

# Estrutura Condicional Composta (cont.)

- Exemplo:

```
if (n1 >= 0)
    printf("Número positivo!");
else
    printf("Número negativo!");
```



# Estrutura Condicional Composta

- Forma 2: bloco de comandos:

```
if (condição)
{
    comando1;
    comando2;
    comando3;
}
else
{
    comando4;
    comando5;
}
```



# Estrutura Condicional Composta e Aninhada

```
if (condicao1)
{
    comando1;
    if (condicao2)
        comando2;
    else
    {
        comando3;
        comando4;
    }
}
```



# Vários if-else aninhados

```
if(condicao1)
    comando 1;
else
    if(condicao2)
        comando 2;
    else
        if(condicao3)
            comando3;
        else
            comando4;
```

# Exemplo: Programa maior número

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ float num1, num2;
  printf("\nDigite dois numeros: ");
  scanf("%f%f",&num1,&num2);
  if (num1 > num2)
    printf("\nO maior numero : %.2", num1);
  else
    if (num2 > num1)
      printf("\nO maior numero : %.2", num2);
    else
      printf("\nOs numeros sao iguais");
  system("Pause"); }
```



# Estrutura Case (Comando Seletivo)

**Atenção:** os tipos de variáveis permitidos são apenas os tipos **int** e **char**

- **Sintaxe:**

```
switch (variável)
{
    case valor1: lista de comandos;
                break;
    case valor2: lista de comandos;
                break;
    ...
    default: lista de comandos;
}
```





# Programa exemplo

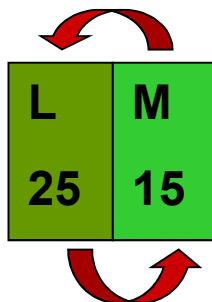
```
#include <stdio.h>
#include <stdlib.h>
int main()
{   int num;
    printf("\nDigite um numero: ");
    scanf("%d", &num);
    switch(num)
    {   case 1: printf("\nNumero = 1");
        break;
        case 2: printf("\nNumero = 2");
        break;
        default: printf("\nNumero diferente de 1 e de 2!");
    }
    system("Pause");
}
```

# Troca de conteúdo entre duas variáveis

memória

L	M
25	15

1ª Idéia:

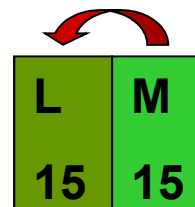


L	M
25	15

$$L = M$$

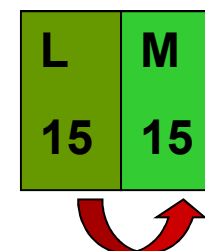
$$M = L$$

$$L = M$$



L	M
15	15

$$M = L$$



L	M
15	15

**NÃO FUNCIONA!**

Para trocar é preciso uma variável auxiliar:

L	M	A
25	15	


memória



# Troca de conteúdo entre duas variáveis

Copio o conteúdo de L em

A:  $A = L$




L	M	A
25	15	25

Atribuo o conteúdo de M à

L:  $L = M$

L	M	A
15	15	25

Atribuo o conteúdo antigo de L, guardado em A, à M:  $M = A$



L	M	A
15	25	25



# Exercícios de Fixação

1- Verdadeiro ou falso:

a)  $1 > 2$

b)  $'a' < 'b'$

c)  $3 == 2$

d)  $'1' == '1'$

e)  $3 >= 2$

f)  $'j' != 'j'$

2 - Identifique se há erros no programa a seguir:

```
Main()
```

```
{
```

```
    int a=1; b=2; c=3;
```

```
    printf("Os numeros são: %d %d %d\n, a, b, c, d);
```

```
}
```



## Exercícios de Fixação (cont.)

3- Como será interpretada a expressão  $x+++y$ ?

a)  $x++ + y$       b)  $x + ++y$

4- Qual é o valor de k?

$j = 3;$

$K = j == 3;$

## Exercícios de Fixação (cont.)

- 5- Dados três valores  $X$ ,  $Y$  e  $Z$ , verifique se eles podem ser os comprimentos dos lados de um triângulo. Se forem, verifique se é um triângulo equilátero, isósceles ou escaleno. Se não formarem um triângulo, escreva uma mensagem informando que não é triângulo. Considere que:
- O comprimento de cada lado do triângulo é menor do que a soma dos outros dois lados.
  - O triângulo equilátero tem três lados iguais.
  - O triângulo isósceles tem o comprimento de dois lados iguais.
  - O triângulo escaleno tem os três lados diferentes.



## Exercícios de Fixação (cont.)

6- Faça um programa para resolver equações do 2º grau. Considere:

$ax^2 + bx + c = 0$  **Obs:** a deve ser diferente de 0

$\Delta = b^2 - 4 * a * c$

Se  $\Delta < 0$ , não existe raiz real

Se  $\Delta = 0$ , existe uma raiz real

$x = (-b) / (2 * a)$

Se  $\Delta > 0$ , existem duas raízes reais

$x_1 = (-b + \text{raiz quadrada de } \Delta) / (2 * a)$

$x_2 = (-b - \text{raiz quadrada de } \Delta) / (2 * a)$



# Casos de Testes do Programa (6)

- Testes

1:  $\langle 1, 1, 1; \text{não tem raízes} \rangle$

2:  $\langle 1, 2, 1; -1 \rangle$

3:  $\langle 1, 0, -4; 2, -2 \rangle$





## Exercícios de Fixação (cont.)

7- Faça um programa que receba dois números e execute uma das operações listadas a seguir, de acordo com a escolha do usuário. Se for digitada uma opção inválida, mostre a mensagem de erro e finalize o programa. As opções são:

1. Primeiro número elevado ao segundo número;
2. Raiz quadrada de cada número;
3. Raiz cúbica de cada número;
4. Produto dos números;

**OBS:** Usar comando de seleção múltipla (**case**)