

# Aula 2:

# Introdução à Linguagem C

*Disciplina:* Fundamentos de Programação

**Prof. Luiz Olmes**

*olmes@unifei.edu.br*



# Nas aulas anteriores...

---

## ▶ **O QUE JÁ ESTUDAMOS?**

- ▶ Algoritmos:
  - ▶ Narrativa
  - ▶ Fluxograma
  - ▶ Pseudocódigo

## ▶ **OBJETIVOS:**

- ▶ Histórico da linguagem C.
- ▶ Execução de programas em C.
- ▶ Edição e compilação de códigos C.
- ▶ Primeiro programa em C.
  - ▶ Função de saída `printf`.
- ▶ Exercícios.

## Linguagem C: Histórico

---

- ▶ A linguagem de programação C foi projetada para permitir grande economia de expressão nos programas, isto é, produzir programas fonte mais compactos.

# Linguagem C: Histórico

---

- ▶ A linguagem de programação C foi projetada para permitir grande economia de expressão nos programas, isto é, produzir programas fonte mais compactos.

## Código C

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("Hello!");
5.     return 0;
6. }
```

## Código Assembly 8086

```
1.         org 100h
2.         mov dx, offset texto
3.         mov ah, 9
4.         int 21h
5.         ret
6. texto db "Hello!"
```

# Linguagem C: Histórico

---

```
1. #include <stdio.h>
2. int main()
3. {
4.     int x;
5.     scanf("%d", &x);
6.     return 0;
7. }
```

Código C

```
1.         org 100h
2.         mov cx, 0h
3. entrada:
4.         mov ah, 1
5.         int 21h
6.         cmp al, 13
7.         jz fim
8.         and al, 0fh
9.         mul cx, 0ah
10.        add cl, al
11.        jmp entrada
12.fim:
13.        ret
```

Assembly

# Linguagem C: Histórico

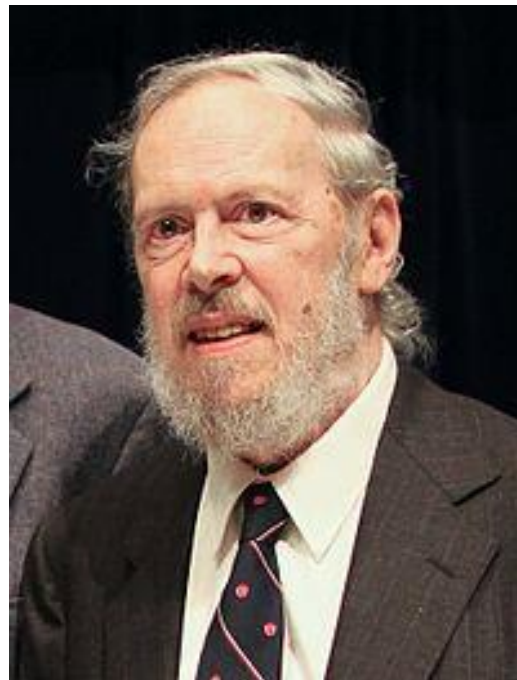
---

- ▶ Foi usada para escrever cerca de 90% do código do sistema operacional Unix.
  - ▶ Com a popularização do Unix em equipamentos de médio porte, e até micros, a linguagem C também ganhou popularidade entre os programadores profissionais.
- ▶ **1969:** os laboratórios Bell lançaram uma versão básica do sistema operacional Unix escrito em Assembly.
  - ▶ Keneth Thompson desenvolveu, em 1969, uma linguagem experimental chamada de linguagem **B**.
- ▶ **1972:** a partir da linguagem B, a linguagem **C** foi projetada.

# Linguagem C: Histórico

---

- ▶ Dennis MacAlistair Ritchie:
  - ▶ ☆ 09/11/1941
  - ▶ † 12/10/2011
  - ▶ Criador da linguagem C (entre outros).



# Linguagem C: Histórico

---

- ▶ **1973:** o sistema operacional Unix foi melhorado, e cerca de 90% de código foi escrito em C.
- ▶ Devido a libertação do Assembly, o Unix (e, conseqüentemente, C), adquiriu grande portabilidade.
  - ▶ C foi rapidamente adaptada a uma série de computadores e seu uso não parou de crescer.
- ▶ Atualmente:
  - ▶ Diversos tipos de aplicações são desenvolvidas em linguagem C, sejam elas voltadas para microcomputadores, dispositivos embarcados ou aparelhos eletrônicos (celulares, tablets, etc).



# Principais linguagens de programação

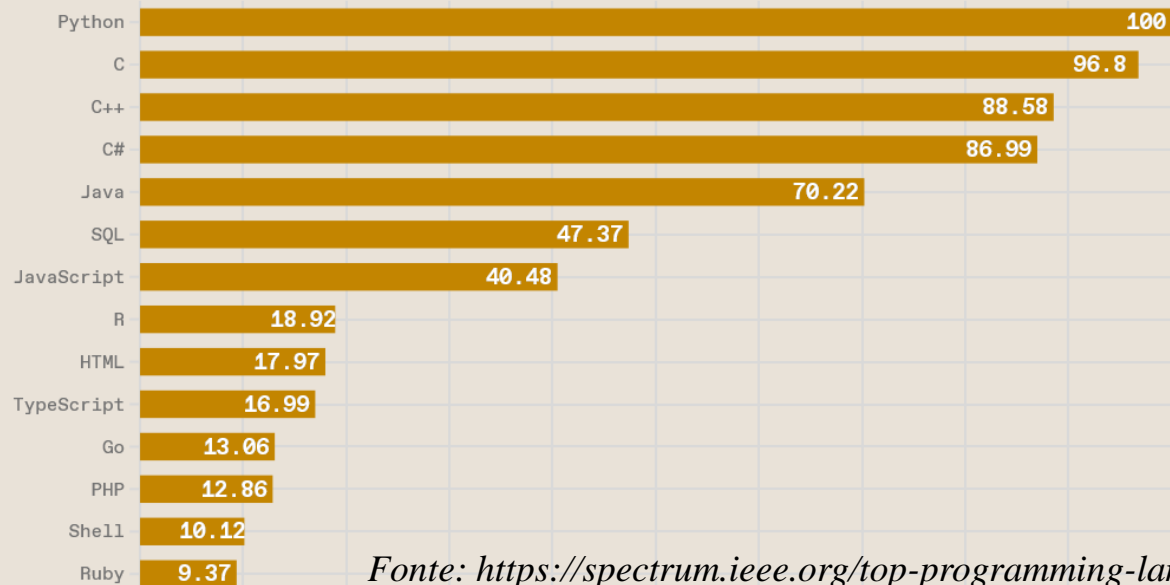
## Top Programming Languages 2022

Click a button to see a differently weighted ranking

Spectrum

Jobs

Trending



Rank de 2022 (o de 2023  
ainda não foi publicado!)

Fonte: <https://spectrum.ieee.org/top-programming-languages/>

# Execução de Programas em C

---



# Execução de Programas em C

---



- ▶ **Edição:**
- ▶ Esta primeira fase consiste na edição do código do programa.
- ▶ É realizada em software de edição de código fonte:
  - ▶ Eclipse, Netbeans, CodeBlocks, CLion, Dev C++, Visual Studio, etc.
- ▶ Ou até mesmo em editores de texto:
  - ▶ Notepad++, Vim, emacs, Kate, etc.
- ▶ Arquivos tem a extensão **.c**.

# Execução de Programas em C

---



- ▶ **Preprocessamento:**
- ▶ O preprocessor C executa as diretivas de preprocessamento:

```
#include  
#define  
#undef  
#line  
#error  
#pragma
```

```
#if  
#ifdef  
#ifndef  
#else  
#elif  
#endif
```

# Execução de Programas em C

---



## ▶ **Compilação:**

- ▶ Fase em que ocorre a tradução do programa para linguagem de máquina.
- ▶ O compilador verifica o código em busca de erros de sintaxe.
- ▶ Se houverem erros, o programa não é traduzido:
  - ▶ Erros de compilação.
- ▶ Caso contrário, é gerado um módulo objeto: extensão **.o**.

# Execução de Programas em C

---



- ▶ **Ligação:**
- ▶ Programas em C normalmente contêm referências para funções definidas externamente, como em bibliotecas.
- ▶ O módulo objeto é gerado sem essas referências.
- ▶ Um software especial chamado **linker** realiza a ligação entre as funções externas e o módulo objeto do programa.
- ▶ Resultado: arquivo executável, com extensão **.exe**.

# Execução de Programas em C

---



- ▶ **Carregamento:**
- ▶ Esta fase é realizada por um software do sistema operacional conhecido como **loader** (carregador).
- ▶ Este software busca o arquivo executável no disco e o transfere para a memória principal.
- ▶ Adicionalmente, também são carregadas bibliotecas utilizadas no programa.

# Execução de Programas em C

---



- ▶ **Execução:**
- ▶ Após estar carregado na memória, o programa é executado.
- ▶ Esta fase ocorre sob o controle da CPU.



# Edição de Código C

---

- ▶ A programação em C pode ser feita em diversos ambientes.
- ▶ Um dos mais simples é o **editor de texto** que acompanha a distribuição Linux instalada na máquina.
  - ▶ Se você usa Windows, instale o Ubuntu em uma máquina virtual para usar nesta disciplina.
- ▶ Basta abrir o editor de texto, salvar o arquivo com a extensão **.c** e iniciar a programação.
- ▶ Ao final, sempre antes de testar o código, lembre-se de salvá-lo.
- ▶ A partir de então, será necessário **compilar** o código para poder testá-lo.

# Primeiro programa em C

---

- ▶ Imprimindo uma linha de texto (**não coloque acentos ou cedilha no código!**):

# Primeiro programa em C

---

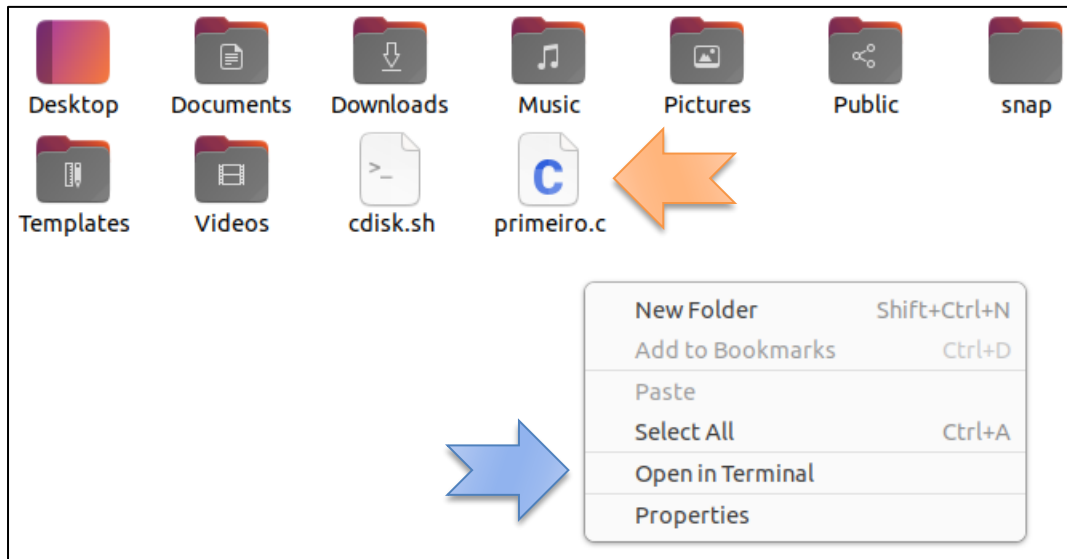
- ▶ Imprimindo uma linha de texto (**não coloque acentos ou cedilha no código!**):

primeiro.c

```
1. // Arquivos de cabeçalho usados no programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```

# Compilando o Código C

- ▶ A compilação do código fonte é realizada no **Terminal**.
- ▶ Através do Gerenciador de Arquivos, navegue até a pasta onde o seu código fonte está salvo.
- ▶ Clique com o botão direito na área branca e selecione a opção “**Abrir no Terminal**”.
- ▶ A telinha preta do Terminal se abrirá.



## Compilando o Código C

---

- ▶ Na janela do Terminal, o seguinte comando compila o arquivo de código fonte:
  - ▶ `gcc -o saida arquivo_fonte.c`

## Compilando o Código C

---

- ▶ Na janela do Terminal, o seguinte comando compila o arquivo de código fonte:

▶ `gcc -o saida arquivo_fonte.c`



compilador C    nome do arquivo    código fonte  
                  executável gerado

# Compilando o Código C

---

- ▶ Na janela do Terminal, o seguinte comando compila o arquivo de código fonte:

▶ `gcc -o saida arquivo_fonte.c`



compilador C    nome do arquivo    código fonte  
                 executável gerado

- ▶ Se o programa apresentar **erros**, eles serão listados logo após a invocação do comando `gcc`.
- ▶ Antecedendo a mensagem de erro, tem-se o número da linha onde o processo de compilação falhou.
  - ▶ `arquivo_fonte.c:L:C`, onde L é o número da linha e C é a coluna.

# Erros mais comuns

---

- ▶ Ausência de final de declaração:
  - ▶ Ocorre quando o programador esquece de terminar a instrução com um ponto e vírgula.
  - ▶ **Error: expected ';' before ...**
- ▶ Uso de variável que não foi declarada:
  - ▶ **Error: 'x' undeclared (first use in this function)**
- ▶ Indefinição da função principal:
  - ▶ Ocorre quando o programador digita o nome da função principal com erro.
  - ▶ **Undefined reference to 'main'**
- ▶ Parênteses, chaves ou colchetes faltantes:
  - ▶ **Error: expected identifier or '(' before '}' token**
  - ▶ **Error: expected declaration or statement at end of input**



## Executando o Código C

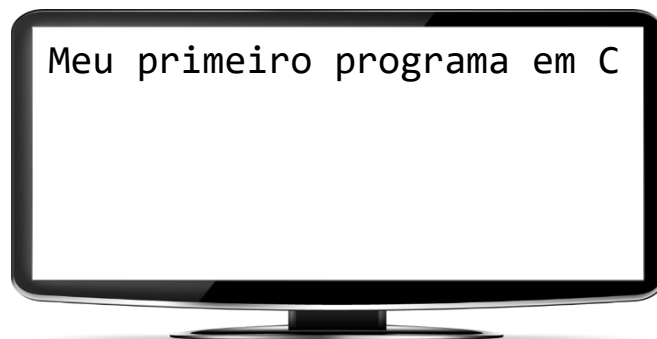
---

- ▶ Após compilado (e não apresentar erros), o programa pode ser executado. Retomando o exemplo anterior:
  - ▶ `gcc -o saida arquivo_fonte.c` (compila)
  - ▶ `./saida` (executa)
- ▶ Invocando simplesmente o nome do arquivo executável gerado (saida), o programa se abrirá para que os dados sejam digitados e as respostas obtidas.
- ▶ Todas as vezes em que o código fonte for alterado, é necessário salvá-lo e compilá-lo novamente.

# Primeiro programa em C: resultado da execução

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```



# Primeiro programa em C: análise

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```

Comentários

Comentários

Comentários

Comentários

# Comentários

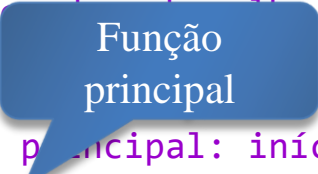
---

- ▶ Utilizados para documentar um programa e aumentar sua legibilidade.
- ▶ Não são executados pelo programa: ignorados pelo compilador.
- ▶ Em C:
  - ▶ Comentário de linha: **// comentário**
    - ▶ Comenta somente a linha atual.
  - ▶ Comentário de bloco: **/\*  
comentário  
\*/**
    - ▶ Comenta um bloco de código (permite pular linhas).

# Primeiro programa em C: análise

---

```
1. // Arquivos e bibliotecas utilizados pelo programa (bibliotecas)
2. #include
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```



Função principal

# Função Principal (main)

---

- ▶ `int main()`
- ▶ Ponto de partida / início de execução de qualquer programa C.
- ▶ Os parênteses após o `main` indicam que é uma função.
- ▶ Programas C geralmente contêm várias funções. Uma delas deve ser, **obrigatoriamente**, a função `main` e deve ser definida como mostrado.
- ▶ O `int` antes do nome da função indica que a função retorna um valor inteiro ao sistema operacional, que indica a existência ou não de erro.

# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Bem-vindo ao mundo em C \n");
9.
10.    // Final que não houve erro
11.    return 0;
12. }
```

A função main é delimitada  
por abre / fecha chaves { }

# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Primeiro programa em C\n");
9.
10.    // Finaliza o programa
11.    return 0;
12. }
```

A função `main` é delimitada por abre / fecha chaves { }

Tudo o que estiver dentro das chaves é chamado de “*corpo*” da função `main`.



# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     TAB // Comando para imprimir uma linha de texto
8.     TAB printf("Meu primeiro programa em C \n");
9.
10.    TAB // Informa o Sistema Operacional que não houve erro
11.    TAB return 0;
12. } // Fim
```

Para facilitar a identificação do corpo da função apenas olhando o código, as instruções são deslocadas à direita por um TAB.

# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```

Instrução: faz com que o computador realize uma ação.

# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```

Instrução: faz com que o computador realize uma ação.

Toda instrução termina com um ponto e vírgula

## Comando de saída

---

- ▶ `printf("Meu primeiro programa em C \n");`
- ▶ A função `printf` é uma das funções de saída da linguagem C.
- ▶ Permite que um programa C exiba informações na janela de comando (terminal, console ou prompt):
  - ▶ No exemplo: `Meu primeiro programa em C`.
- ▶ A sequência de caracteres entre os parênteses é sempre **delimitada por aspas duplas** e é chamada de **argumento** da função `printf`.

## Comando de saída

---

- ▶ `printf("Meu primeiro programa em C \n");`
- ▶ No final do argumento da função `printf`, está presente um caractere especial: `\n`.
- ▶ Este caractere faz com que uma linha seja pulada.
- ▶ Experimente adicionar outro `\n` no argumento:
  - ▶ `printf("Meu primeiro\nprograma em C \n");`
  - ▶ e reexecutar o código.

## Comando de saída

---

- ▶ Experimente adicionar mais `\n` no argumento:
- ▶ `printf("Meu\nprimeiro\nprograma\nem\nC \n");`
- ▶ e reexecutar o código.

## Comando de saída

---

- ▶ Experimente adicionar mais `\n` no argumento:
  - ▶ `printf("Meu\nprimeiro\nprograma\nem\nC \n");`
  - ▶ e reexecutar o código.
- ▶ Agora, experimente adicionar um `\t`, da seguinte forma:
  - ▶ `printf("Meu\n\tprimeiro\nprograma\n\tem\nC \n");`
  - ▶ e reexecutar o código.
- ▶ O que faz um `\t`?

## Arquivos de cabeçalho

---

- ▶ A função `printf` está definida em um dos muitos `arquivos de cabeçalho` (*header*) da linguagem C (também chamados de bibliotecas).
- ▶ Para que seja possível utilizar a função `printf`, deve-se incluir o arquivo de cabeçalho que contém sua definição no código.
- ▶ No caso da função `printf`, este arquivo chama-se `stdio.h`.
- ▶ Por isso, no início do código fonte, antes da função `main`, tem-se o comando `#include <stdio.h>`.



# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```

Inclusão do arquivo `stdio.h`.  
STD = standard (padrão)  
IO = input/output (entrada/saída)  
Extensão `.h` vem de *header*

# Primeiro programa em C: análise

---

```
1. // Arquivos de cabeçalho utilizados pelo programa (bibliotecas)
2. #include <stdio.h>
3.
4. // Função principal: início da execução do programa
5. int main()
6. {
7.     // Comando para imprimir uma linha de texto
8.     printf("Meu primeiro programa em C \n");
9.
10.    // Informa o Sistema Operacional que não houve erro
11.    return 0;
12. } // Fim
```

A instrução return devolve um valor ao sistema operacional. O valor zero indica que está tudo OK.

# Dúvidas?

---



# Estude e pratique!

---

- ▶ Venha para a Aula 03 (*próxima semana*) sabendo **tudo** o que foi visto na Aula 02.



# Aula 2:

# Introdução à Linguagem C

*Disciplina:* Fundamentos de Programação

**Prof. Luiz Olmes**

*olmes@unifei.edu.br*



## Prática: não precisa entregar ao professor

---

- ▶ **Exercício 1:** escreva um código em C que, fazendo uso da função `printf`, imprima os seguintes versos (**sem acentos / cedilha**), respeitando parágrafos, sendo um `printf` para cada verso:

Por mim se vai das dores à morada,  
Por mim se vai ao padecer eterno,  
Por mim se vai à gente condenada.

Moveu Justiça o Autor meu sempiterno,  
Formado fui por divinal possança,  
Sabedoria suma e amor supremo.

No existir, ser nenhum a mim se avança,  
Não sendo eterno, e eu eternal perduto:  
Deixai, ó vós que entraís, toda a esperança!

Estas palavras, em letreiro escuro,  
Eu vi, por cima de uma porta escrito.  
“Seu sentido” – disse eu – “Mestre me é duro”

## Prática

---

- ▶ **Exercício 2:** escreva um código em C que, fazendo uso da função `printf`, imprima os seguintes versos (**sem acentos / cedilha**), respeitando tabulações e parágrafos:

Às parelhas das carruagens do Faraó  
eu te comparo, minha amada.

Graciosas são tuas faces entre os brincos,  
e teu pescoço entre colares.

Faremos para ti brincos de ouro  
com filigranas de prata.

## Prática

---

- ▶ **Exercício 3:** escreva um código em C que, fazendo uso da função printf, imprima:
  - ▶ a) A mensagem “*This is a C program!*” em uma única linha.
  - ▶ b) Usando dois printf, a mensagem “*This is a C program!*”, onde *C program* fica na segunda linha.
  - ▶ c) Usando um único printf, a mensagem “*This is a C program!*”, onde *C program* fica na segunda linha.
  - ▶ d) Usando um único printf, a mensagem “*This is a C program!*”, sendo que cada palavra deve ficar em uma linha.
  - ▶ e) Usando um único printf, a mensagem “*This is a C program!*”, sendo que cada palavra deve ficar separada da outra por um TAB.



# Prática

---

- ▶ **Exercício 4:** escreva um código em C que, fazendo uso da função printf, imprima os seguintes versos (**sem acentos / cedilha**), lado a lado:

Como dois e dois são quatro  
Sei que a vida vale a pena  
Embora o pão seja caro  
E a liberdade pequena

Como um tempo de alegria  
Por trás do terror me acena  
E a noite carrega o dia  
No seu colo de açucena

Como teus olhos são claros  
E a tua pele, morena  
como é azul o oceano  
E a lagoa, serena

- sei que dois e dois são quatro  
sei que a vida vale a pena  
mesmo que o pão seja caro  
e a liberdade pequena.

# Prática

- **Exercício 5:** escreva um programa em C que, fazendo uso da função printf, imprima um retângulo, uma oval, uma seta e um losango, como mostrado abaixo (uma figura ao lado da outra):

```

*****          ****              *                *
*               *           *             ***        *   *
*               *           *             *****     *   *
*               *           *             *            *       *
*               *           *             *            *       *
*               *           *             *            *       *
*               *           *             *            *       *
*               *           *             *            *       *
*               *           *             *            *       *
*****          ****              *                *

```

## Prática

---

- ▶ **Exercício 6:** escreva um programa em C que, fazendo uso da função `printf`, imprima as iniciais de seu nome em letras de imprensa (*block letters*).  
Exemplo: Carlos Silva ficaria:

CCCCCCCC	SSSSSSSS
CC	SS
CC	SSSSSSSS
CC	SS
CCCCCCCC	SSSSSSSS

## Prática

---

- **Exercício 7:** escreva um código em C que, fazendo uso da função `printf`, imprima a imagem de um tabuleiro de xadrez, como mostrado a seguir:

```
      a b c d e f g h
-----
8 |T|C|B|Q|K|B|C|T| 8
7 |P|P|P|P|P|P|P|P| 7
6 | | | | | | | | 6
5 | | | | | | | | 5
4 | | | | | | | | 4
3 | | | | | | | | 3
2 |P|P|P|P|P|P|P|P| 2
1 |T|C|B|Q|K|B|C|T| 1
-----
      a b c d e f g h
```