

**Período:** 1º Semestre

**Disciplina:** Algoritmos e Lógica de Programação

**Professor:** Profa. Dra. Ligia Rodrigues Prete

**E-mail:** [ligia.prete@fatec.sp.gov.br](mailto:ligia.prete@fatec.sp.gov.br)

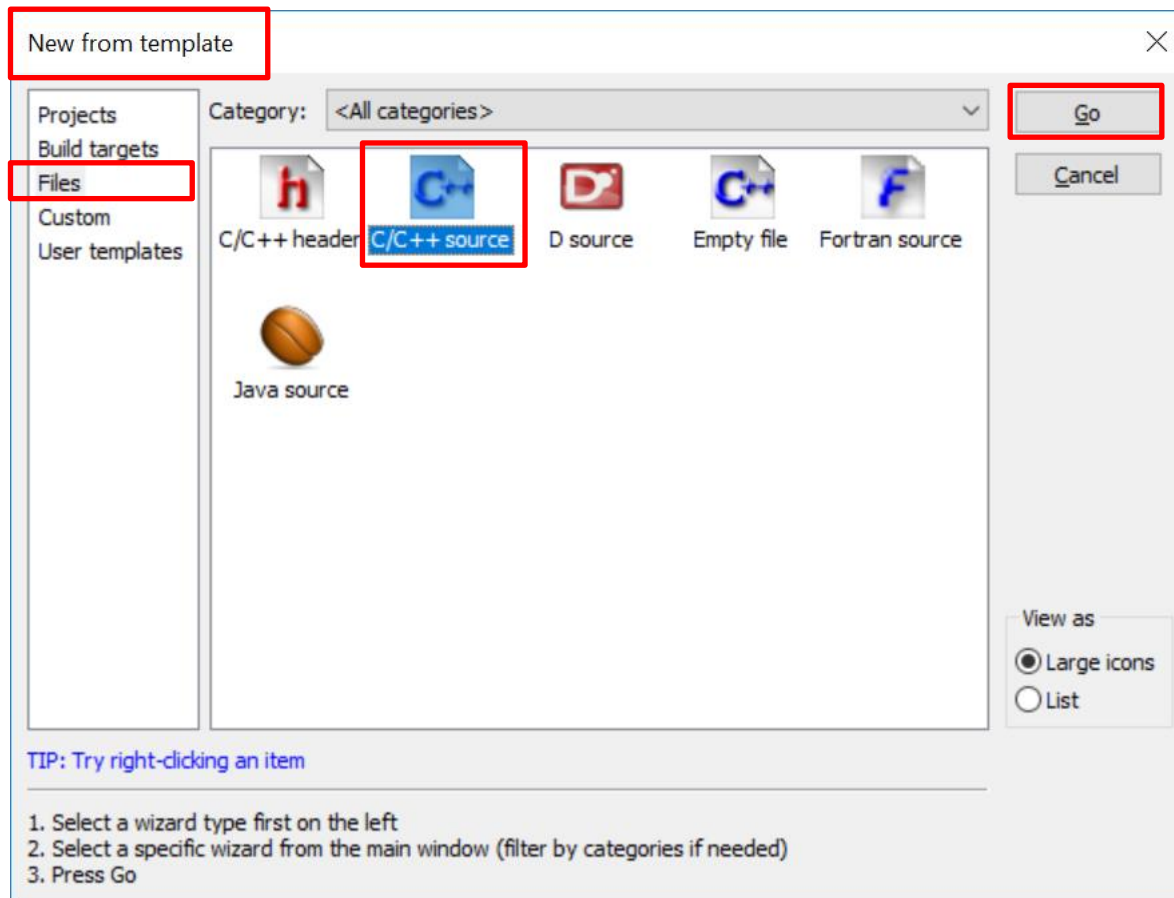
**4 – Linguagem C**

# Um pouco de história

- A linguagem C foi criada em 1972 por Dennis Ritchie no *Bell Telephone Laboratories*;
- É uma das linguagens mais populares;
- É compilada;
- Existem poucas arquiteturas de hardware que não oferecem suporte a C;
- Influenciou diversas linguagens como Java, PHP, Pascal, entre outras;
- C é uma linguagem estruturada e C++ é orientada a objetos.

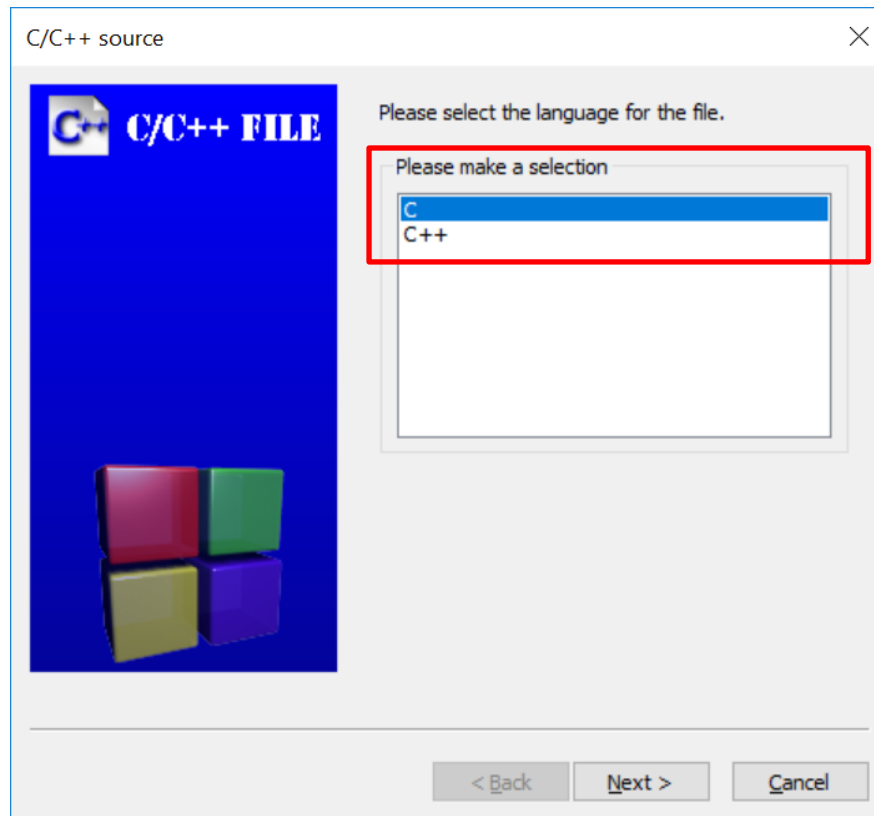
# Criando um projeto C/C++ no Code::Blocks

- Abra o **Code:Blocks**;
- Clique no menu “**File>New>File**”;
- Na janela “**New from template**”, selecione “**Files**” (lado esquerdo);
- Na direita, selecione “**C/C++ source**”. Clique no botão “**Go**”.



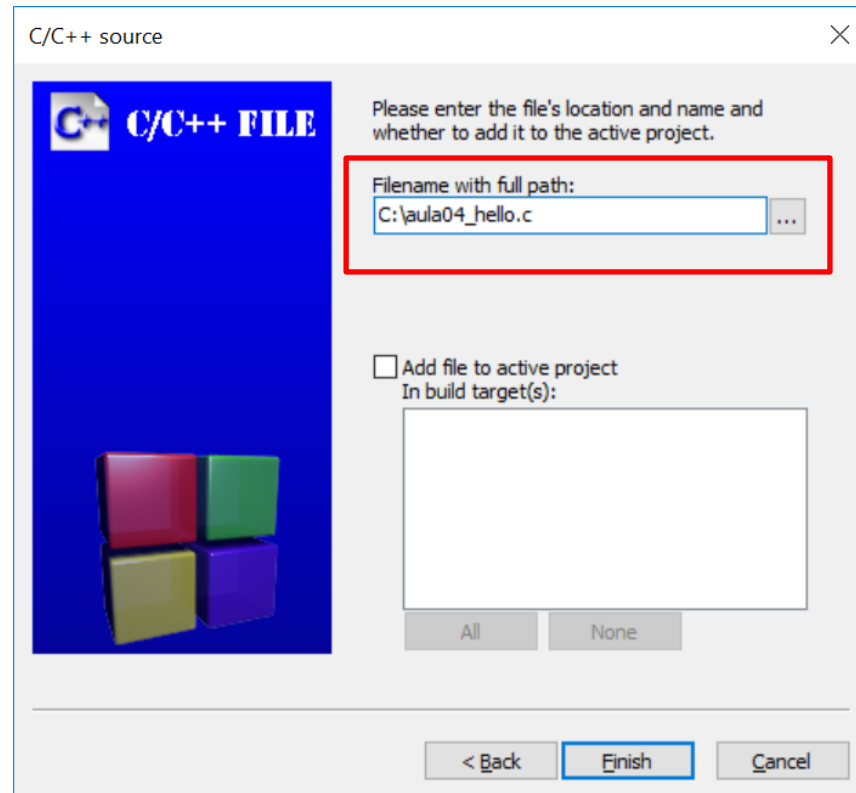
# Configurando o projeto

- Na janela “C/C++ source” selecione em “Please make a selection” a linguagem “C”.
- Clique em “Next”.



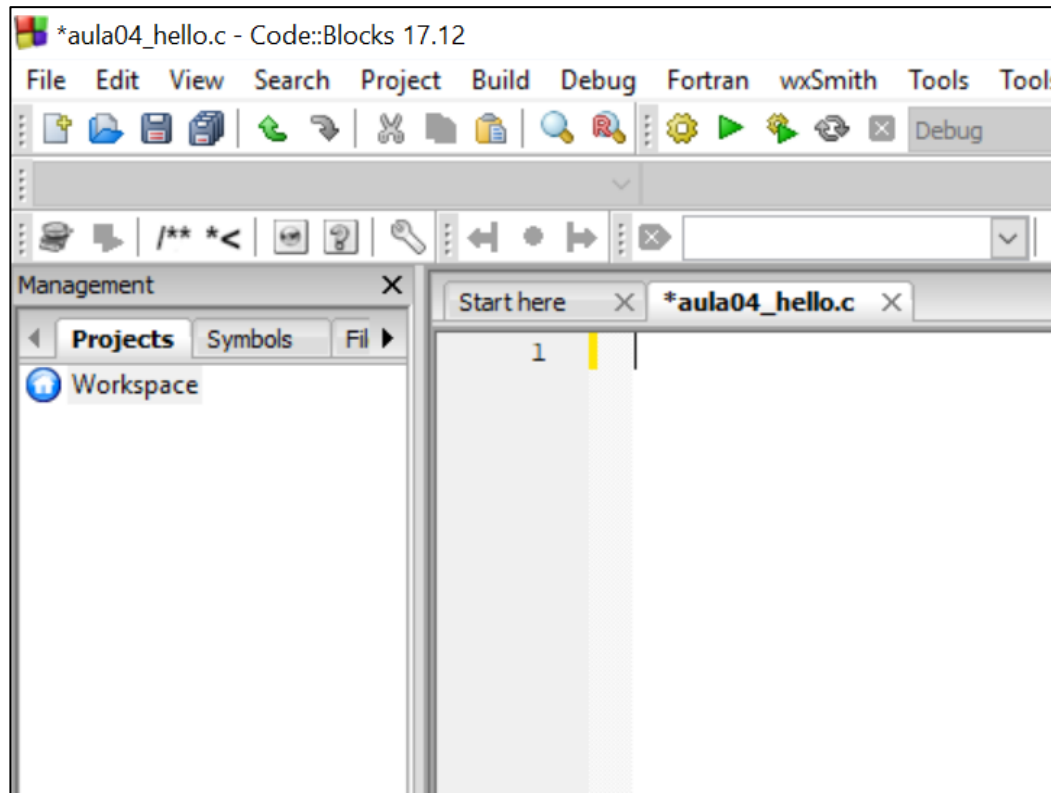
# Configurando o projeto

- Na janela “**C/C++ source**” selecione em “**Filename with full path:**” o nome do arquivo código-fonte e o local.
- Clique em “**Finish**”.



# Configurando o projeto

- O arquivo código-fonte “**aula04\_hello.c**” aparece vazio.



# As chaves – { }

- As chaves possuem um papel importante nas linguagens de programação modernas. Elas indicam blocos de comandos, isto é, marcam o início e o fim de blocos de comandos.
- São usadas para marcar início e fim de programas, estruturas condicionais, estruturas de repetição, classes, métodos e outros elementos que fazem parte do código. Exemplo:

**estrutura ou programa {**

Comando 01

Comando 02

...

...

Comando N

**}**

# O sinal ; (ponto e vírgula)

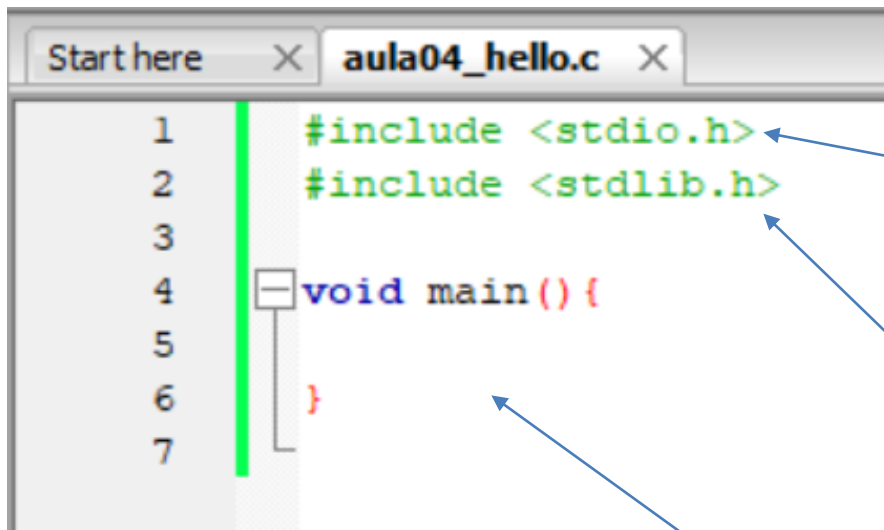
- A cada final de linha de código sempre há o sinal “;” (ponto e vírgula).
- Esse sinal indica o fim da instrução e deve ser colocado em todo final de linha que envolva algum processamento direto de dados tais como atribuição, operações matemática, etc.
- Se esquecer de colocar o sinal “;” nas linhas corretas o programa não será compilado, pois possui erros.



# Linguagem C é Case Sensitive

- A linguagem C é **Case Sensitive**, isto é, faz diferenciação entre maiúsculas e minúsculas;
- Não é a mesma coisa escrever **main()**, **Main()**, **MAIN()** ou **mAIN()**;
- Todas as instruções de C são escritas com letra minúscula;
- Só deve utilizar letras maiúsculas quando desejar-se utilizar nome de variáveis, mensagens ou funções escritas pelo programador.

# Código-fonte básico do arquivo “aula04\_hello.c”



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main() {
5
6  }
7
```

Diretiva que inclui as funções de entrada (printf) e saída (scanf)

Diretiva que inclui as funções dos comandos do DOS. Por exemplo:  
system("calc");

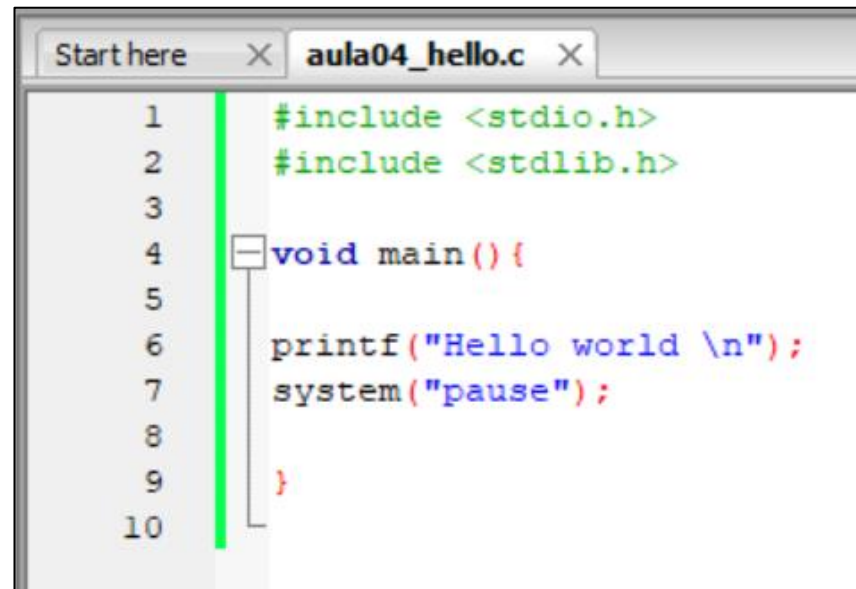
Método principal do programa (main), a lógica deverá ser codificada aqui dentro.

# Bibliotecas de Funções

- A linguagem C não possui, por exemplo, comandos de escrita e leitura. Para resolver o problema existe um conjunto de funções na Biblioteca de Funções.
- No exemplo anterior do **main.c** aparecem duas diretivas que indica ao compilador para adicionar ao processo de compilação os arquivos:  
**stdio.h (standard input/output)**  
**stdlib.h (standard library)**

# Código-fonte do arquivo “aula04\_hello.c”

- Uma das funções que permite a escrita na tela é a função **printf** (**print** + **formatted** – escrita formatada).
- O caractere especial **\n** serve para indicar que a próxima escrita será na linha abaixo.



The image shows a screenshot of a code editor window with two tabs: 'Start here' and 'aula04\_hello.c'. The code in the 'aula04\_hello.c' tab is as follows:

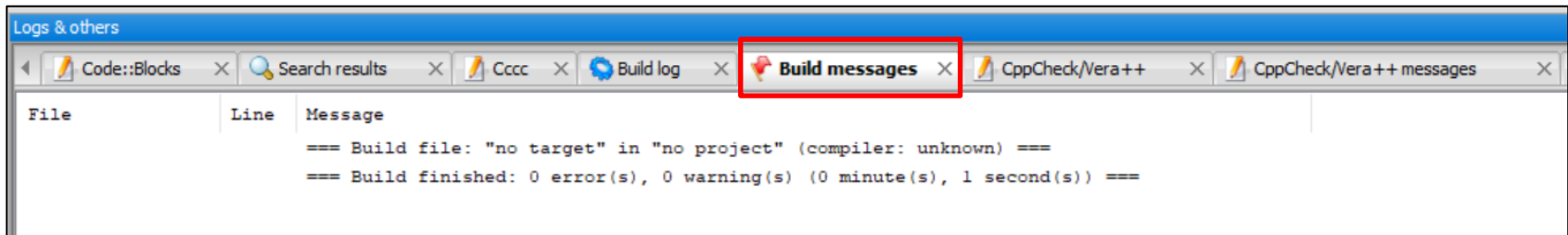
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main() {
5
6      printf("Hello world \n");
7      system("pause");
8
9  }
10
```

# Verificando Logs e Mensagens



The screenshot shows the 'Build log' window in Code::Blocks. The window title is 'Logs & others'. The tab bar contains several tabs: 'Code::Blocks', 'Search results', 'Cccc', 'Build log' (highlighted with a red box), 'Build messages', 'CppCheck/Vera++', and 'CppCheck/Vera++ messages'. The main text area displays the following build output:

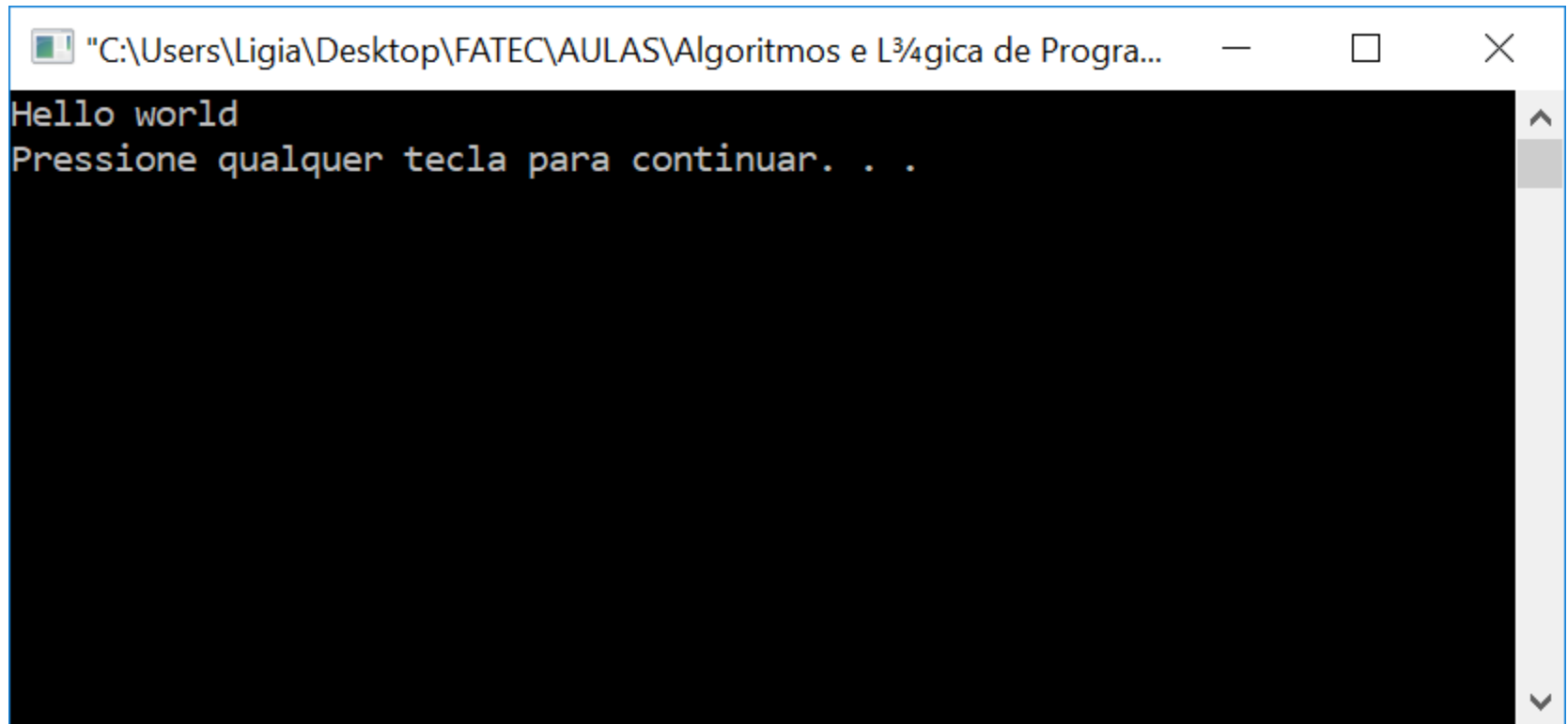
```
----- Build file: "no target" in "no project" (compiler: unknown)-----  
mingw32-gcc.exe -c C:\Temp\aula04_hello.c -o C:\Temp\aula04_hello.o  
mingw32-g++.exe -o C:\Temp\aula04_hello.exe C:\Temp\aula04_hello.o  
Process terminated with status 0 (0 minute(s), 1 second(s))  
0 error(s), 0 warning(s) (0 minute(s), 1 second(s))  
  
Checking for existence: C:\Temp\aula04_hello.exe  
Executing: '"C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Temp\aula04_hello.exe"' (in 'C:\Temp')  
Process terminated with status 0 (0 minute(s), 7 second(s))
```



The screenshot shows the 'Build messages' window in Code::Blocks. The window title is 'Logs & others'. The tab bar contains several tabs: 'Code::Blocks', 'Search results', 'Cccc', 'Build log', 'Build messages' (highlighted with a red box), 'CppCheck/Vera++', and 'CppCheck/Vera++ messages'. The main text area displays the following build messages:

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
		=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 1 second(s)) ===

# Visualizando o resultado no Terminal



A screenshot of a Windows terminal window. The title bar at the top shows the file path "C:\Users\Ligia\Desktop\FATEC\AULAS\Algoritmos e Lógica de Progra..." followed by standard window control buttons (minimize, maximize, close). The terminal area has a black background with white text. The first line displays "Hello world". The second line displays "Pressione qualquer tecla para continuar. . .". A vertical scrollbar is visible on the right side of the terminal window.

```
"C:\Users\Ligia\Desktop\FATEC\AULAS\Algoritmos e Lógica de Progra...  
Hello world  
Pressione qualquer tecla para continuar. . .
```

# **Ciclo de Desenvolvimento de uma Aplicação 1/4**

- **Edição do código-fonte**

Nesta fase, todo o trabalho é realizado pelo programador, o qual deverá escrever o código-fonte em arquivos com extensão “.c”

# Ciclo de Desenvolvimento de uma Aplicação 2/4

- **Compilação do programa**

Uma vez feito o programa, entra o processo de **compilação** que é realizado pelo **compilador**. Caso seja detectado algum erro, o processo de compilação é terminado. O programador terá que corrigir o erro encontrado pelo compilador.

Caso não seja detectado nenhum erro, o compilador cria um arquivo objeto com extensão **“.o”**, com o mesmo nome do arquivo **“.c”**



# Ciclo de Desenvolvimento de uma Aplicação 3/4

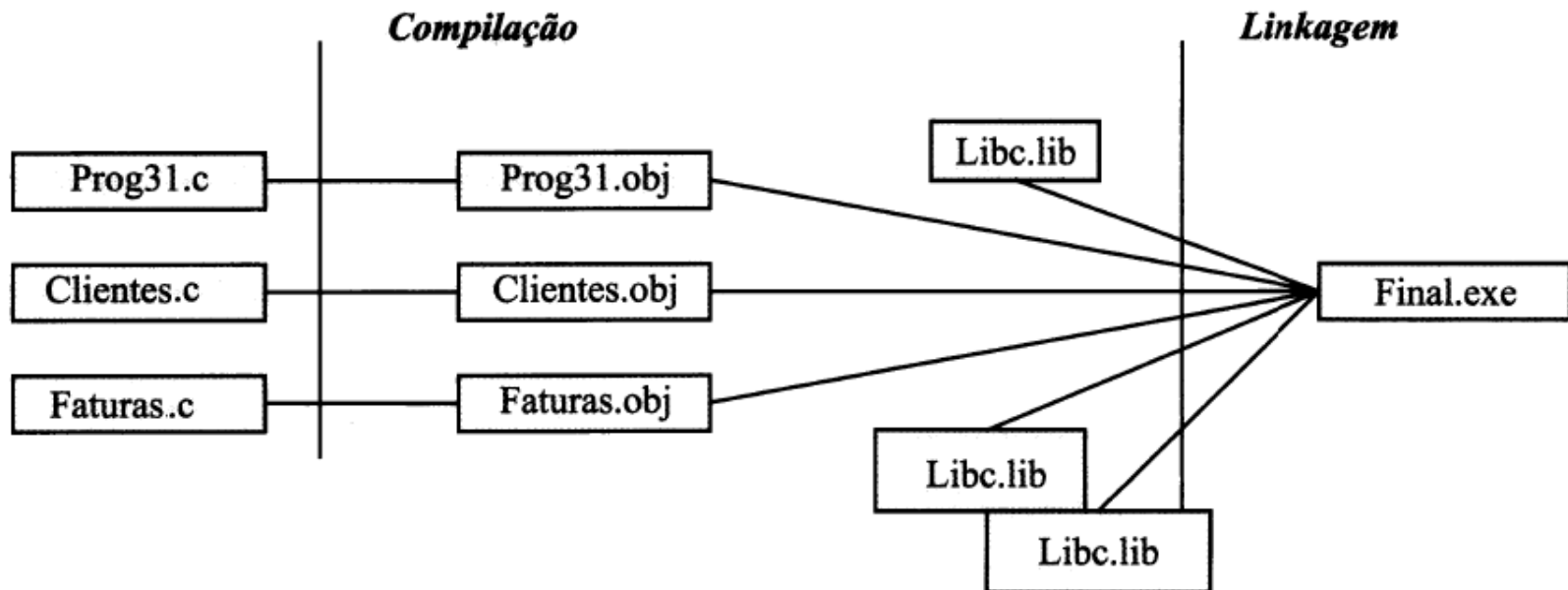
- Linkagem dos objetos

O arquivo executável “**.exe**” é criado a partir da linkagem dos arquivos objetos e das bibliotecas da linguagem C.

# Ciclo de Desenvolvimento de uma Aplicação 4/4

- Execução do programa

Se o processo de linkagem terminar com sucesso, tem-se disponível um arquivo executável.



# Code::Blocks

- É uma IDE utilizada para compilar programas na Linguagem C, encontrado no link:

<http://www.codeblocks.org/downloads/26>

- Escolha o arquivo “**codeblocks-17.12mingw-setup.exe**” que já possui o compilador GCC/G++