Representação de Algoritmos

Bruno A. N. Travençolo – FACOM-UFU

Formas de Representação de Algoritmos

A descrição de um algoritmo de **forma clara e fácil de ser seguida** ajuda no seu desenvolvimento, depuração (localização e correção de erros) e futura migração para uma linguagem de programação.

Para facilitar este trabalho, são utilizadas ferramentas específicas de representação da lógica de programação (*seqüência de ações a serem realizadas*).

Linguagens de Programação

- Linguagem de programação é um conjunto de termos (vocabulário) e de regras (sintaxe) que permitem a formulação de instruções a um computador
  - Linguagem de máquina: nível mais baixo, elementar. Utiliza-se apenas 0 e 1

```
00010000100011100010110000111001
10101001010010100101111 1
0101010100101010 1
```

Linguagens de Programação

– Linguagem simbólica (assembly): utiliza códigos mnemônicos (abreviações para as instruções ao invés de números). Esses códigos são traduzidos para instruções de máquina executáveis pelo computador

– Exemplo

Instruction (AT&T syntax)
.begin
.org 2048
.equ 3000
ld length,%
be done
addcc %r1,-4,%r1
addcc %r1,%r2,%r4
ld %r4,%r5
ba loop
addcc %r3,%r5,%r3
jmpl %r15+4,%r0
20
a_start
.org a_start

Linguagem Assembly p/ ling. de Máquina

Endereço	Rótulo	Instruction (AT&T syntax)	Código objeto (linguagem de máquina)
		.begin	
		.org 2048	
	a_start	.equ 3000	
2048		ld length,%	
2064		be done	00000010 10000000 00000000 00000110
2068		addcc %r1,-4,%r1	10000010 10000000 01111111 11111100
2072		addcc %r1,%r2,%r4	10001000 10000000 01000000 00000010
2076		ld %r4,%r5	11001010 00000001 00000000 00000000
2080		ba loop	00010000 10111111 11111111 11111011
2084		addcc %r3,%r5,%r3	10000110 10000000 11000000 00000101
2088	done:	jmpl %r15+4,%r0	10000001 11000011 11100000 00000100
2092	length:	20	00000000 00000000 00000000 00010100
2096	address:	a_start	00000000 00000000 00001011 10111000
		.org a_start	
3000	a:		

Linguagens de Programação

- Linguagens de alto nível: Uma linguagem de programação procedural, orientada a problemas, mais próxima a linguagem humana e longe do código de máquina.
  - Ex: C; C++; Java; PASCAL; Fortran; etc
  - Programa em C:

```c
#include<stdio.h>
void torre(int n, char source, char dest,char aux);

main()
{
      int n;
      char source = 'A';
      char dest = 'C';
      char aux = 'B';
      printf("entre com o numero de discos: ");
      scanf("%d",&n);
      torre(n, source,dest,aux);
      getchar();
      getchar();
}

void torre(int n, char source, char dest,char aux)
{
     static int step = 0;

     printf("torre(%d, %c, %c, %c)\n",n, source, dest,aux);
     if (n==1)
       printf("\t\t\tstep %3d: mova de %c para %c\n",++step, source, dest);
     else
```

C para Assembly

```c
#include<stdio.h>
void torre(int n, char source, char dest,char aux);

main()
{
      int n;
      char source = 'A';
      char dest = 'C';
      char aux = 'B';
      printf("entre com o numero de discos: ");
      scanf("%d",&n);
      torre(n, source,dest,aux);
      getchar();
      getchar();
}

void torre(int n, char source, char dest,char aux)
{
     static int step = 0;

     printf("torre(%d, %c, %c, %c)\n",n, source, dest,aux);
     if (n==1)
       printf("\t\t\tstep %3d: mova de %c para %c\n",++step, so
     else
```

```
00401190 50                push eax
00401191 68C7204000        push $004020c7
00401196 E8AB020000        call $00401446
0040119B 83C408            add esp,$08
Unit1.c.12: torre(n, source,dest,aux);
0040119E 8A55F9            mov dl,[ebp-$07]
004011A1 52                push edx
004011A2 8A4DFA            mov cl,[ebp-$06]
004011A5 51                push ecx
004011A6 8A45FB            mov al,[ebp-$05]
004011A9 50                push eax
004011AA FF75FC            push dword ptr [ebp-$04]
004011AD E848000000        call torre(int,signed char,signed
004011B2 83C410            add esp,$10
Unit1.c.13: getchar();
004011B5 8B151C514000      mov edx,[$0040511c]
004011BB FF4A08            dec dword ptr [edx+$08]
004011BE 780A              js $004011ca
004011C0 8B0D1C514000      mov ecx,[$0040511c]
004011C6 FF01              inc dword ptr [ecx]
004011C8 EB0C              jmp $004011d6
004011CA FF351C514000      push dword ptr [$0040511c]
004011D0 E82F020000        call $00401404
004011D5 59                pop ecx
```

Compilador

- Conjunto de programas que transforma uma linguagem de alto nível em linguagem de máquina executável por computador

```
#include<stdio.h>
void torre(int n, char source, char dest,char aux);

main()
{
      int n;
      char source = 'A';
      char dest = 'C';
      char aux = 'B';
      printf("entre com o numero de discos: ");
      scanf("%d",&n);
      torre(n, source,dest,aux);
      getchar();
      getchar();
}

void torre(int n, char source, char dest,char aux)
{
     static int step = 0;

     printf("torre(%d, %c, %c, %c)\n",n, source, dest,aux);
     if (n==1)
       printf("\t\t\tstep %3d: mova de %c para %c\n",++step, source, dest);
     else
```

```
00010000100011100010111001
10101001010010100101111
0101010100101010
```

Linguagem C

Linguagem C

- Desenvolvida em 1972 por **Dennis Ritchie** nos laboratórios *Bell Telephone* para uso com o sistema operacional Unix

Dennis Ritchie (direita) com Ken Thompson

Dados obtidos: Wikipedia

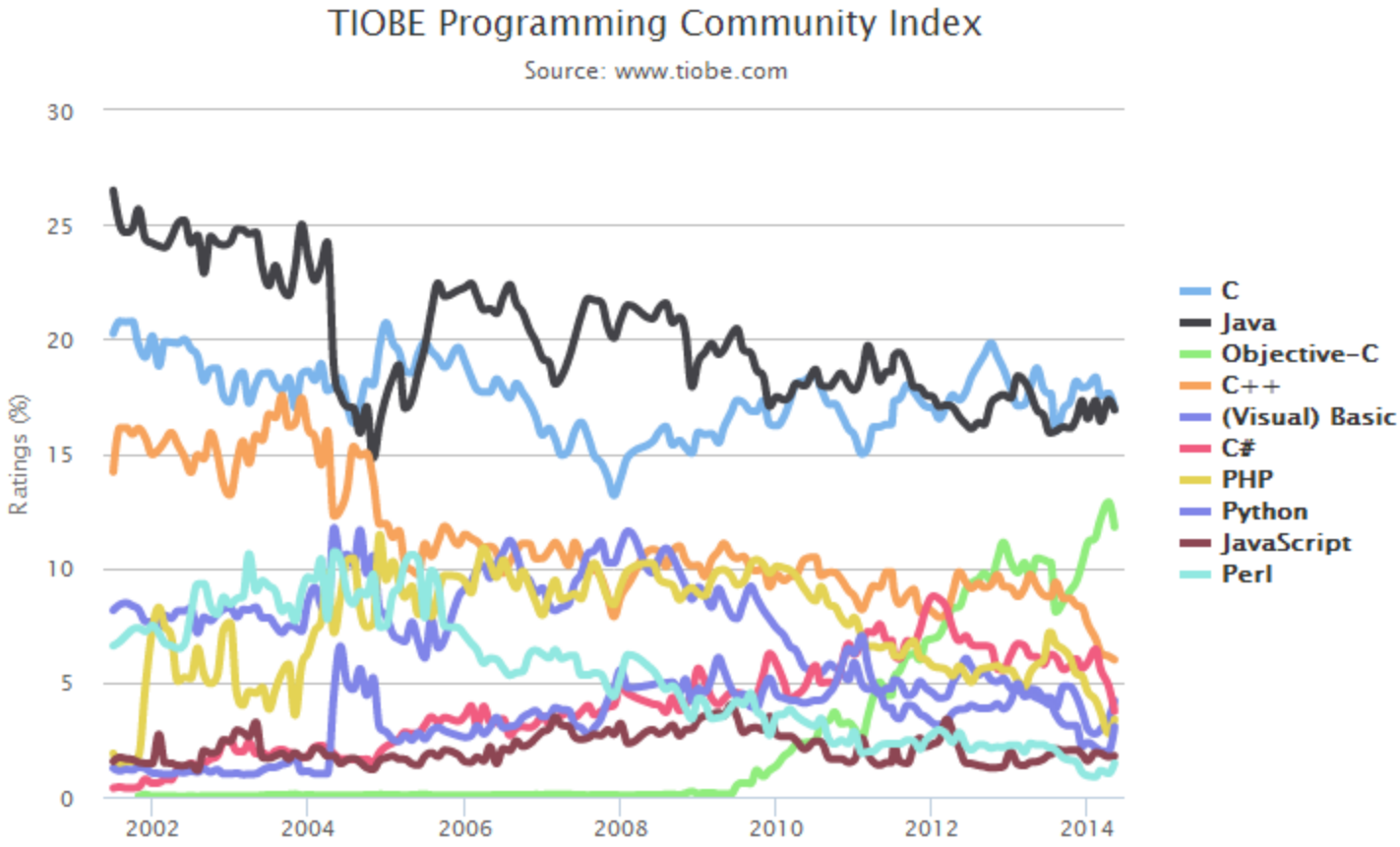
Linguagem C

- Ainda é uma das linguagens mais populares

- Observe that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

May 2014	May 2013	Change	Programming Language	Ratings	Change
1	1		C	16.926%	-1.80%
2	2		Java	16.907%	-0.01%
3	3		Objective-C	11.791%	+1.36%
4	4		C++	5.986%	-3.21%
5	7	↑	(Visual) Basic	4.197%	-0.46%
6	5	↓	C#	3.745%	-2.37%
7	6	↓	PHP	3.386%	-2.40%
8	8		Python	3.057%	-1.26%
9	11	↑	JavaScript	1.788%	+0.25%
10	9	↓	Perl	1.470%	-0.81%
11	12	↑	Visual Basic .NET	1.264%	+0.13%
12	10	↓	Ruby	1.242%	-0.43%
13	38	⇈	F#	1.030%	+0.79%
14	14		Transact-SQL	1.025%	+0.21%
15	17	↑	Delphi/Object Pascal	0.974%	+0.24%
16	13	↓	Lisp	0.967%	+0.07%
17	19	↑	Assembly	0.773%	+0.14%
18	15	↓	Pascal	0.752%	-0.05%
19	21	↑	MATLAB	0.711%	+0.12%
20	42	⇈	ActionScript	0.674%	+0.47%

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
C
Java
Objective-C
C++
(Visual) Basic
C#
PHP
Python
JavaScript
Perl

Linguagem C

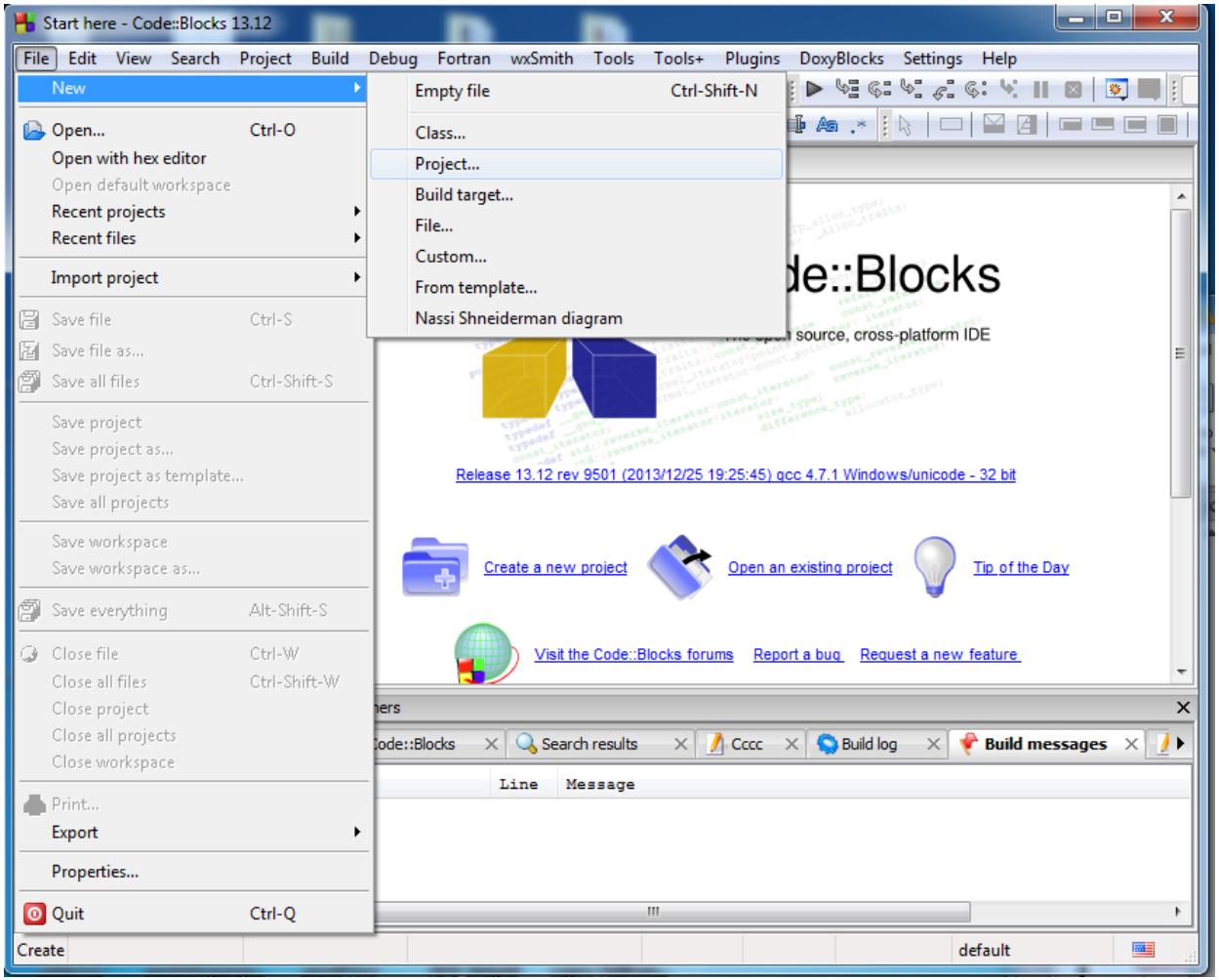
- Em C todos os módulos de um programa são denominados **funções**.
- A função correspondente ao programa ou módulo principal tem o nome de **main** e tem presença obrigatória em qualquer programa.

Utilizando o Code::Blocks

O que é o Code::Blocks

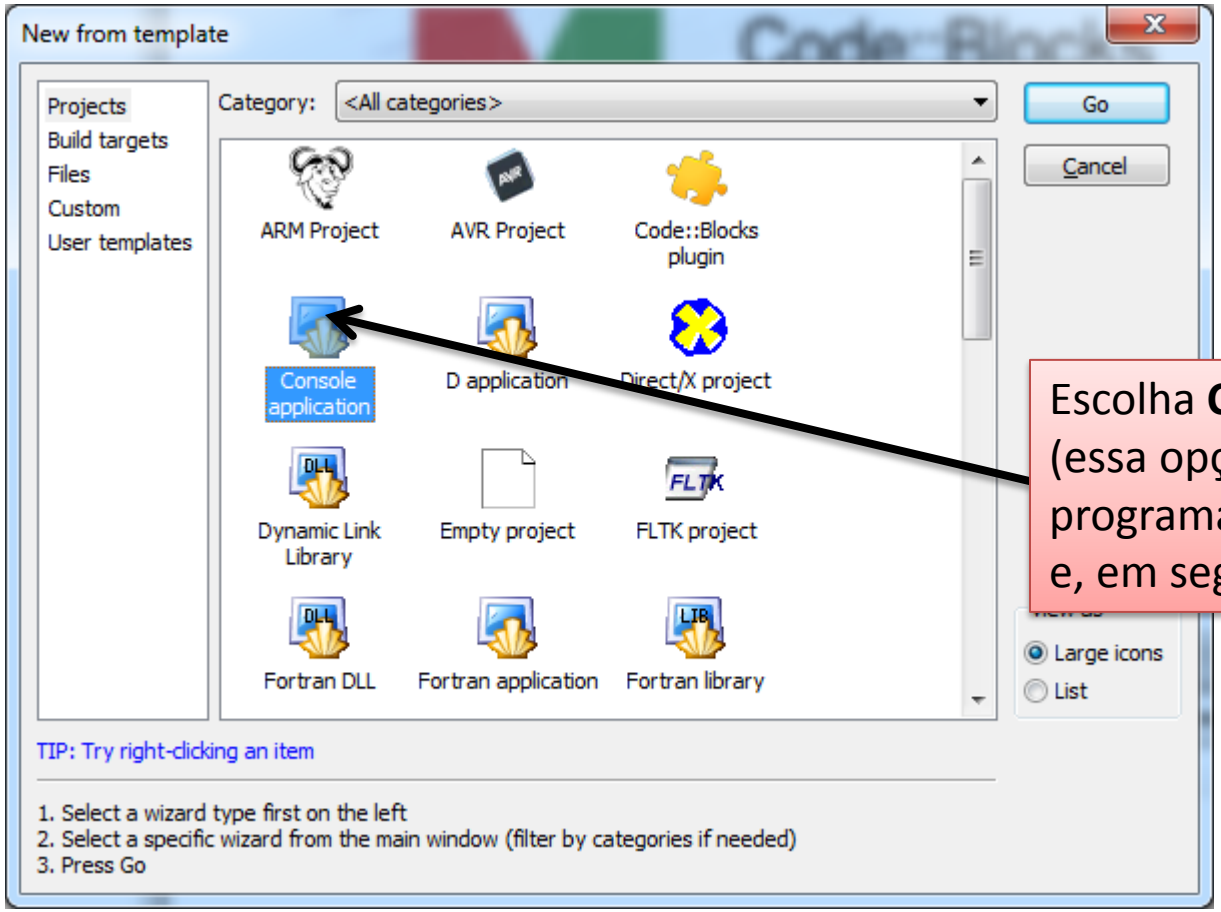
- Programa para desenvolvimento de programas em C (e outras linguagens). É um tipo de software conhecido como IDE (Integrated Development Environment - Ambiente de Desenvolvimento Integrado), pois reúne diversas ferramentas que facilitam a programação.
- Existem outras IDEs que podem ser usadas
  - DevC++, CodeLite, Eclipse, C++Builder, MS Visual Studio, Kdevelop, etc.

Criando um novo projeto



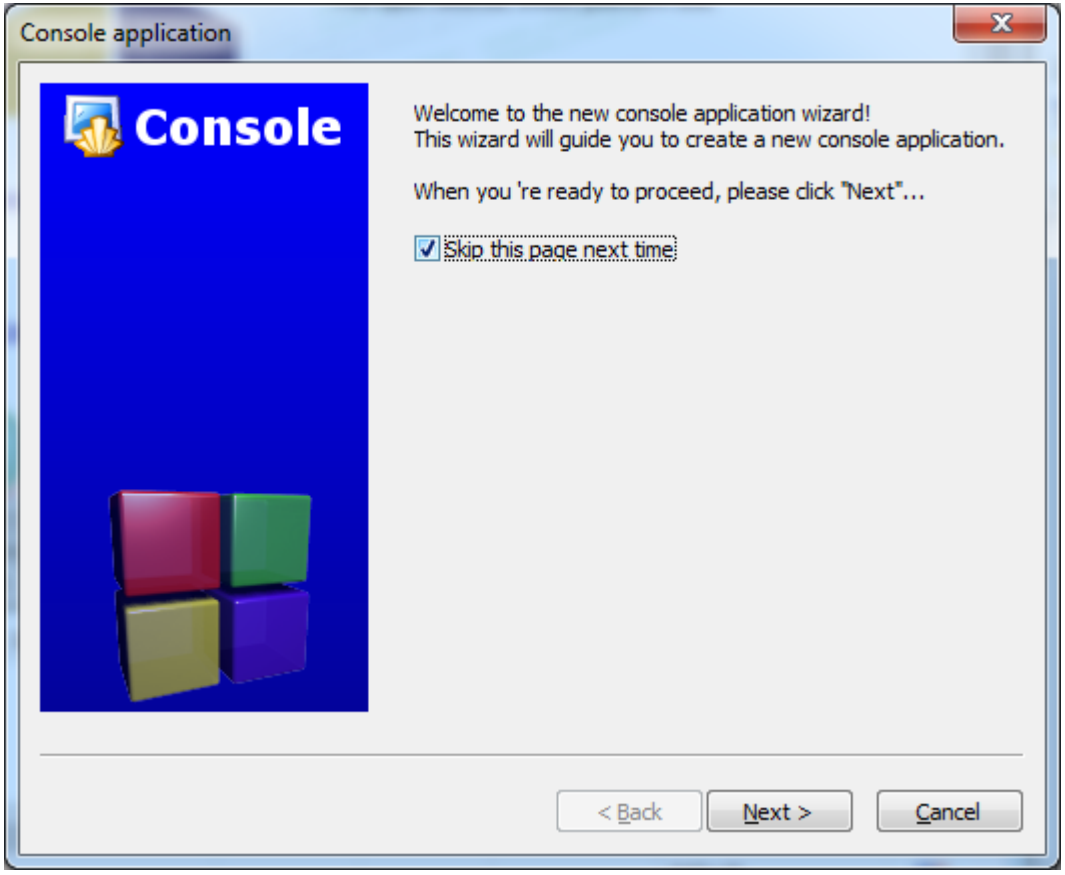
- Clique no menu **File/ New / Project**

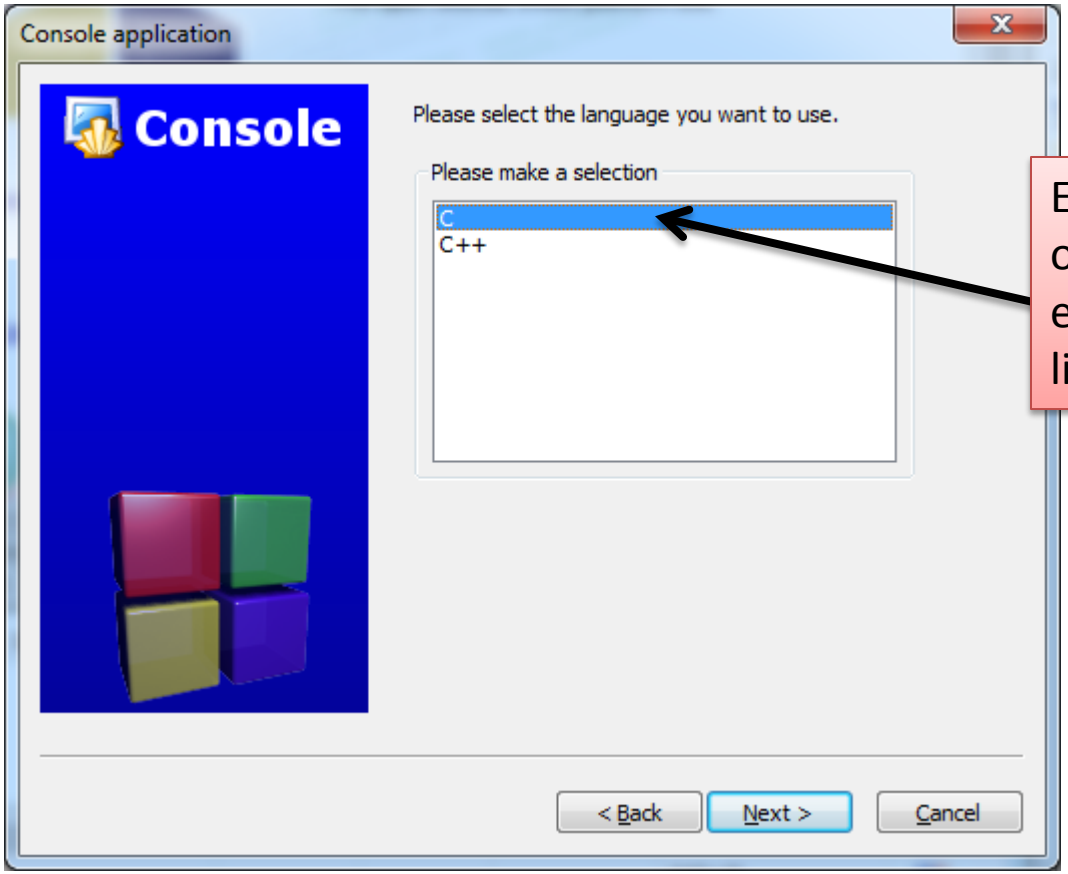
Criando um novo projeto



Escolha **Console Application** (essa opção vai criar um programa sem interface gráfica) e, em seguida, pressione **GO**

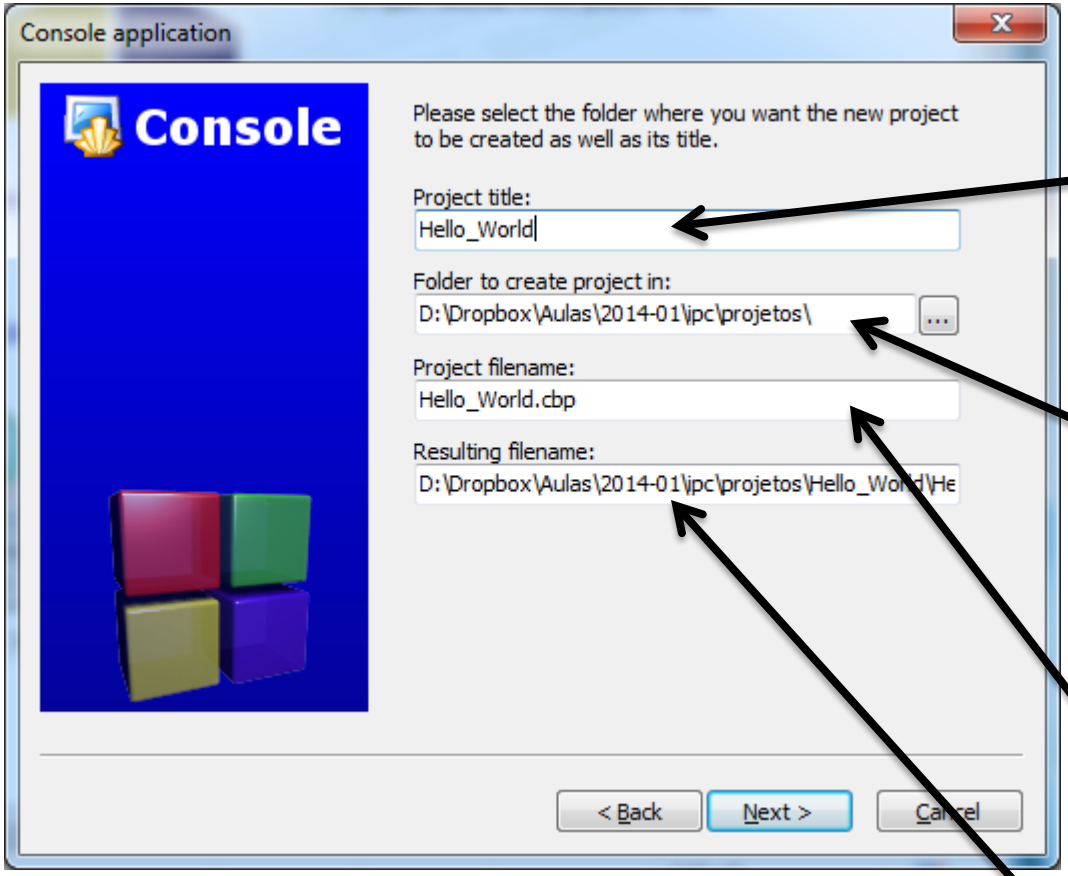
Criando um novo projeto





Escolha a linguagem **C** (essa opção indica que vamos trabalhar em linguagem C e não na linguagem C++).

Criando um novo projeto

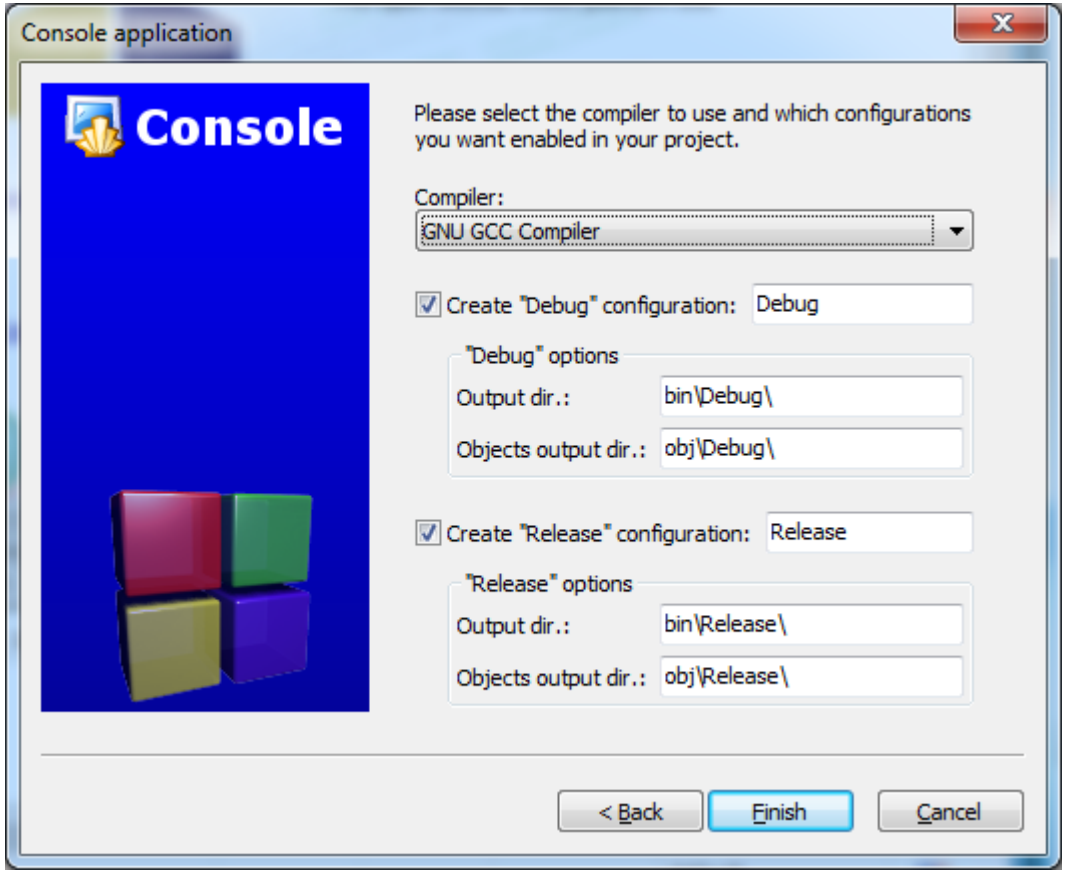


Dê um nome para o seu projeto (Apesar de permitido, evite nome com espaços)

Indique a Pasta (ou diretório) de trabalhos

Nome do arquivo que guardará as informações do projeto

Caminho completo com o arquivo do projeto



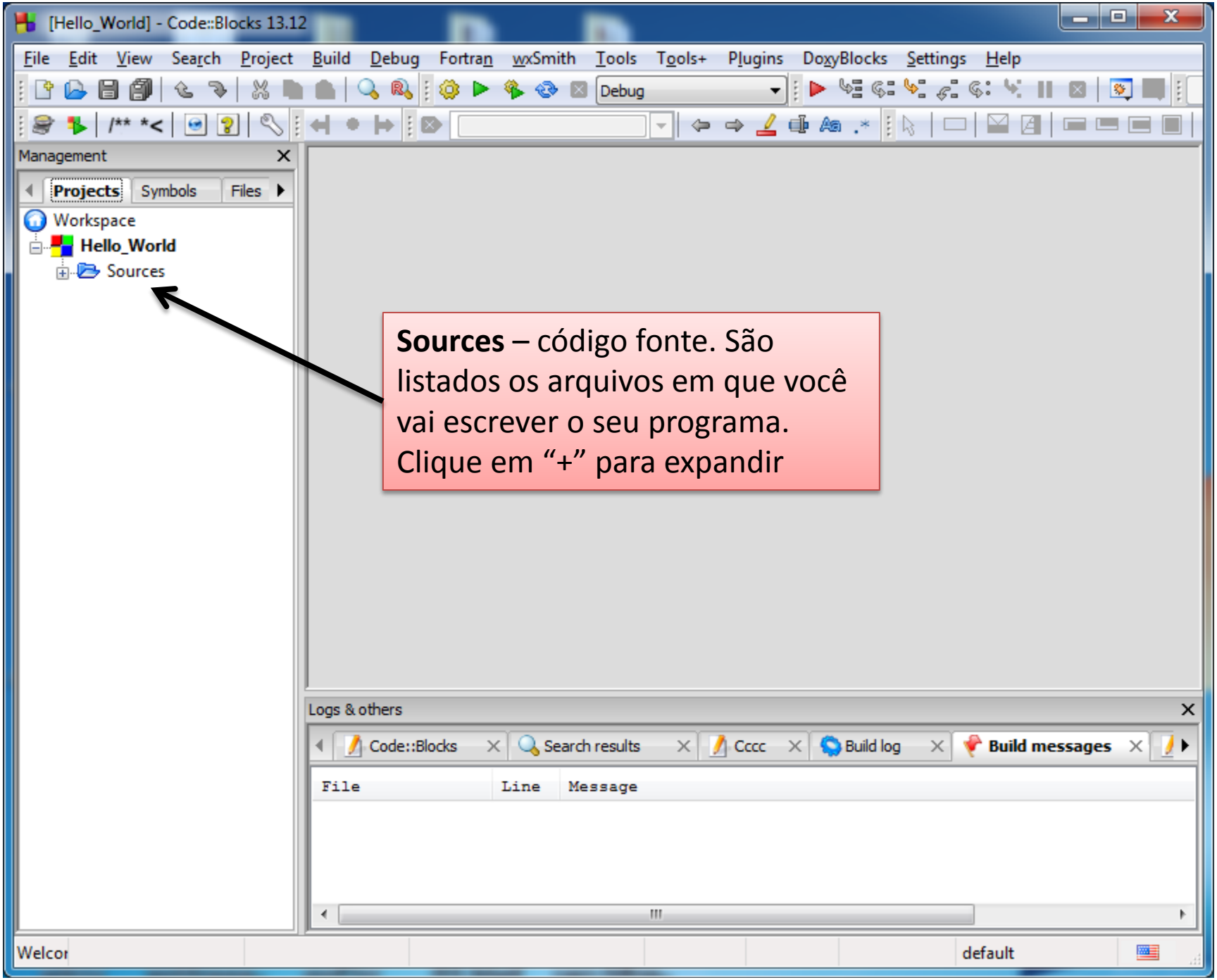
Selecione o compilador padrão (GCC)

Duas opções:

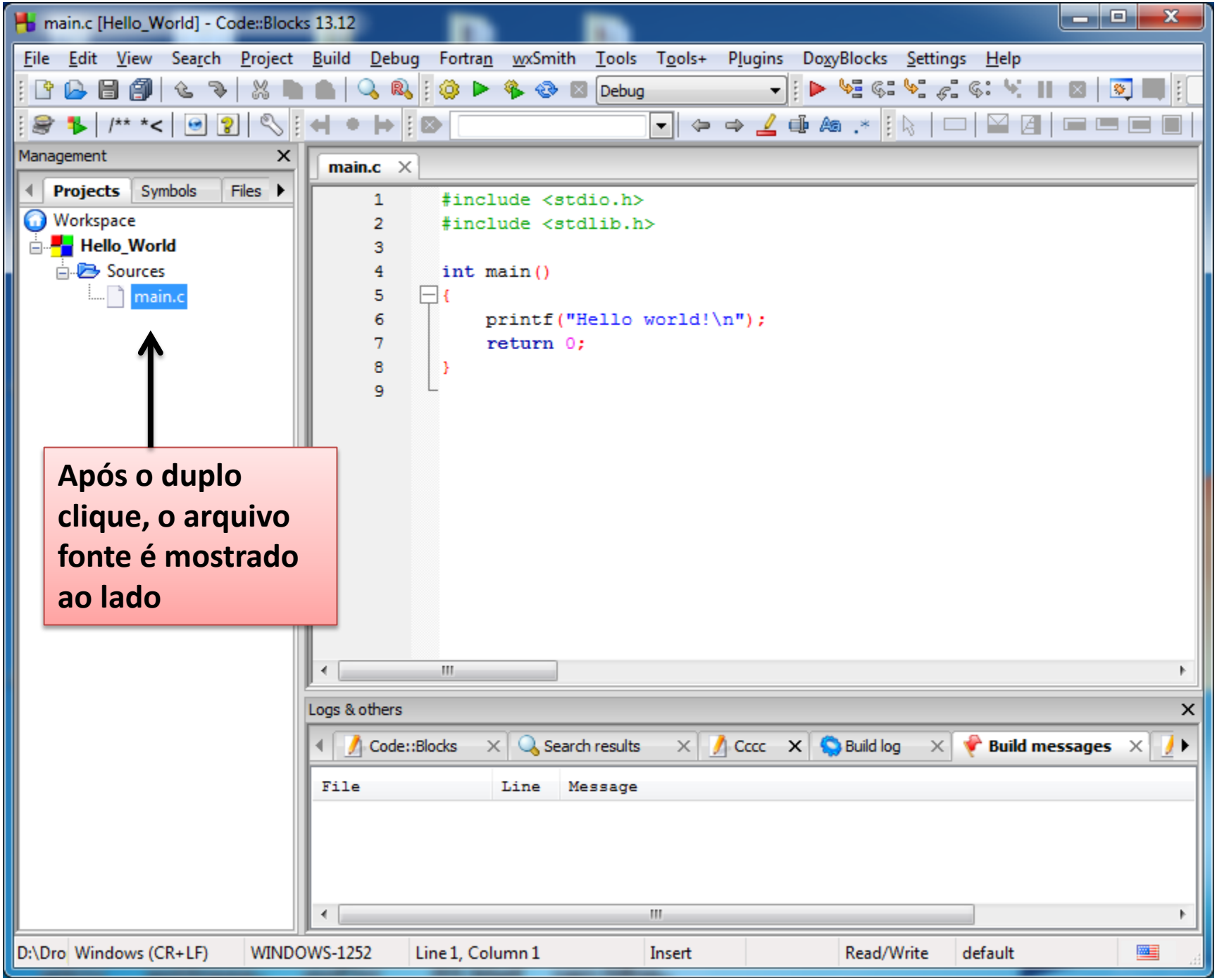
Debug – para versão de testes

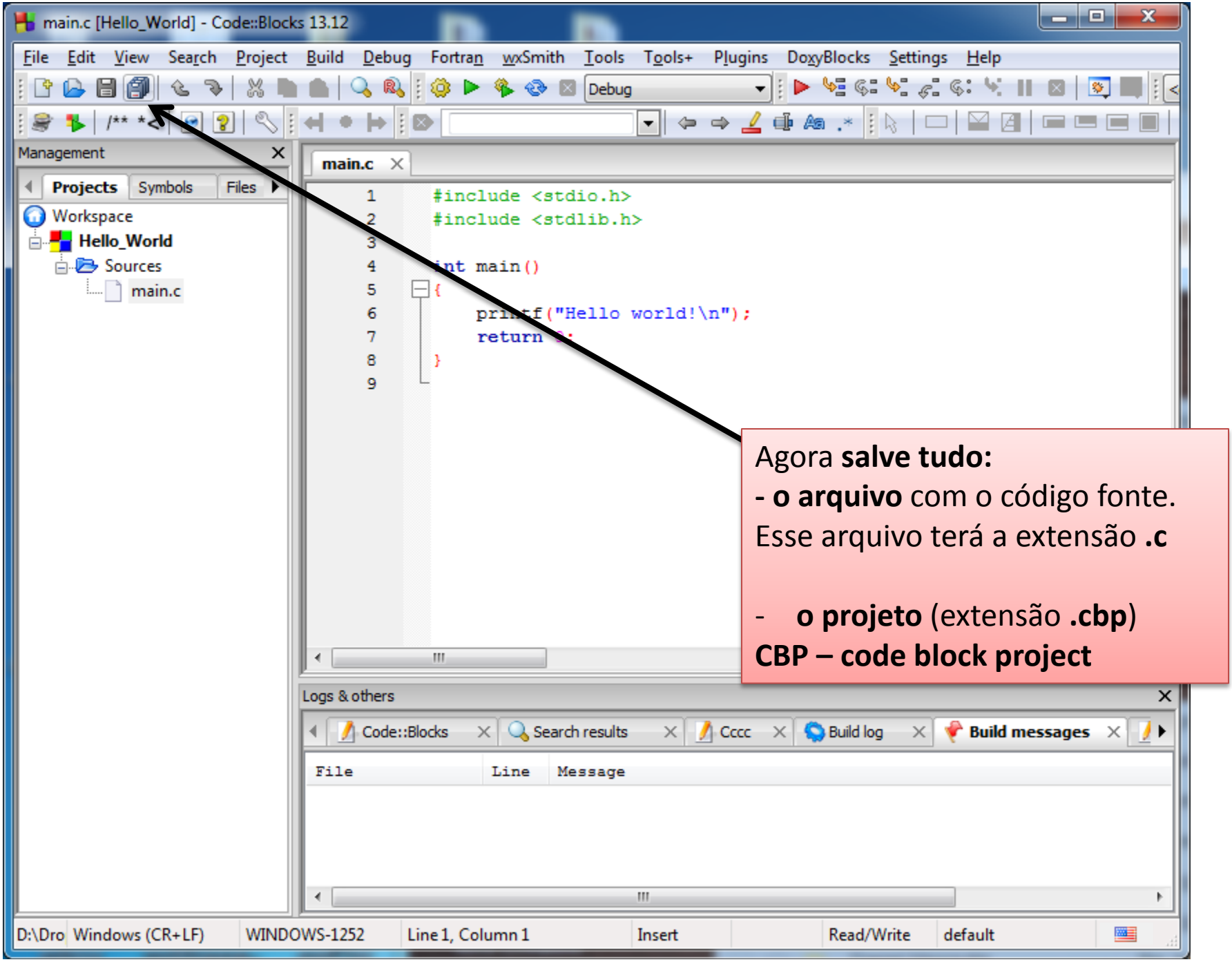
Release – para versão final

** Esses conceitos serão apresentados ao longo do curso

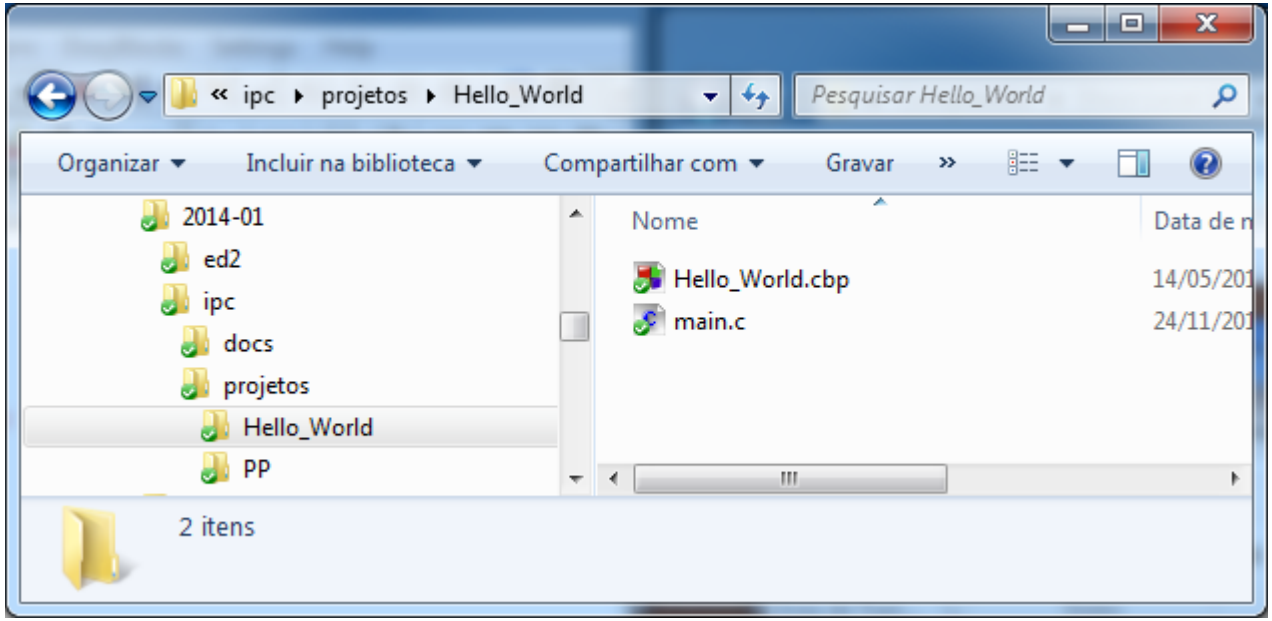


Sources – código fonte. São listados os arquivos em que você vai escrever o seu programa. Clique em "+" para expandir


main.c [Hello_World] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Symbols Files
Workspace
Hello_World
Sources
main.c
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6 printf("Hello world!\n");
7 return 0;
8 }
9
Logs & others
Code::Blocks Search results Cccc Build log Build messages
File Line Message
D:\Dro Windows (CR+LF) WINDOWS-1252 Line 1, Column 1 Insert Read/Write default
Após o duplo clique, o arquivo fonte é mostrado ao lado


main.c [Hello_World] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Symbols Files
Workspace
Hello_World
Sources
main.c
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6 printf("Hello world!\n");
7 return 0;
8 }
9
Logs & others
Code::Blocks Search results Cccc Build log Build messages
File Line Message
D:\Dro Windows (CR+LF) WINDOWS-1252 Line 1, Column 1 Insert Read/Write default
Agora **salve tudo:**
- o arquivo com o código fonte.
Esse arquivo terá a extensão **.c**
- **o projeto** (extensão **.cbp**)
CBP – code block project

Criando um novo projeto



** Verifique se seu Windows oculta as extensões dos arquivos

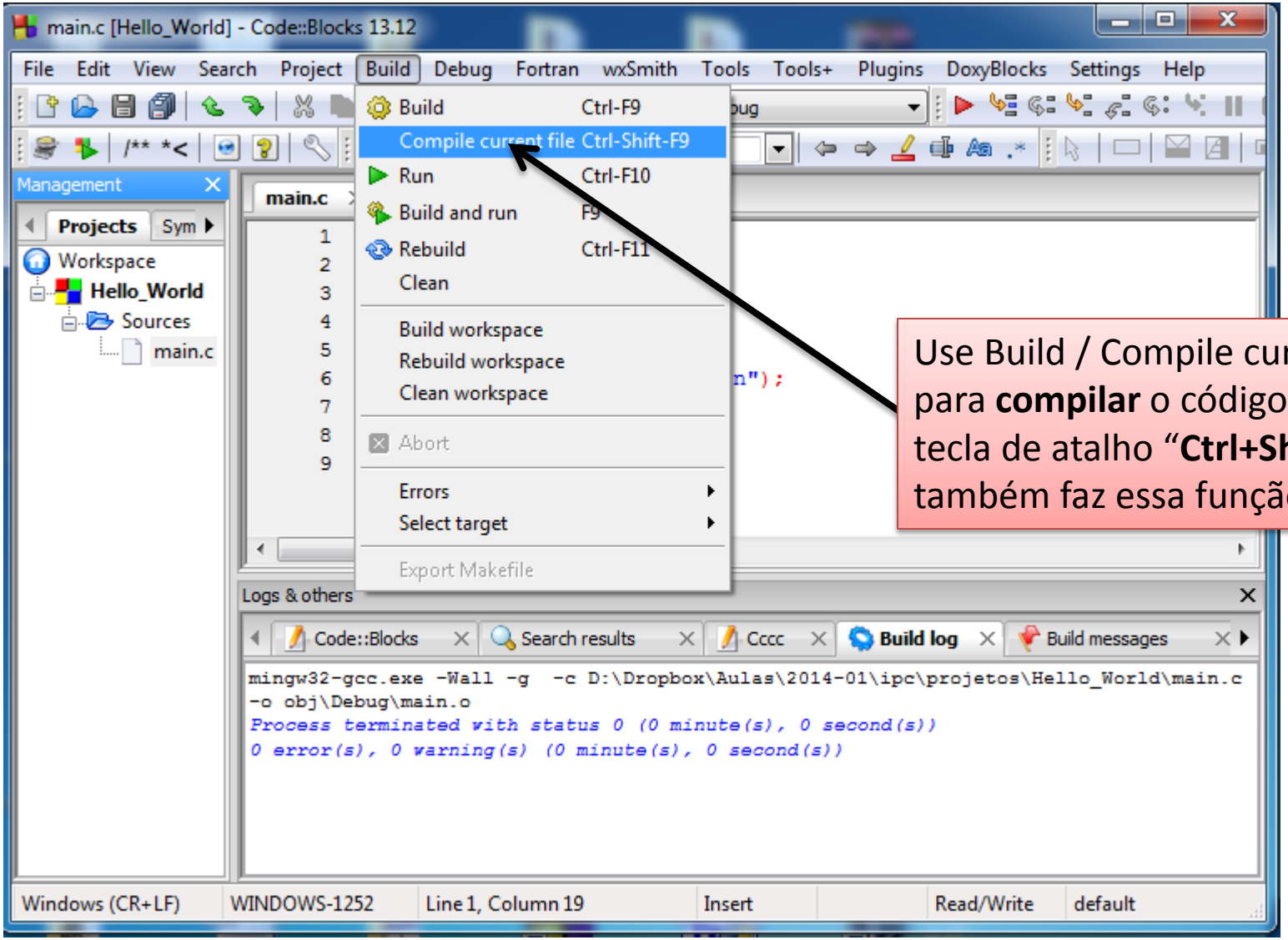
Pressione Alt / Ferramentas / Opções de Pasta / Modo de Exibição / "Ocultar extensão de arquivos conhecidos"

Depois de salvo, a pasta em que o código foi salvo contém dois arquivos:

(i) Um com a extensão **.cbp**, que armazena dados do projeto em Dev C++

(ii) Outro com a extensão **.c**, que armazena o código fonte do programa que está sendo desenvolvido

Compilando o código fonte



(relembrando...) Compilador

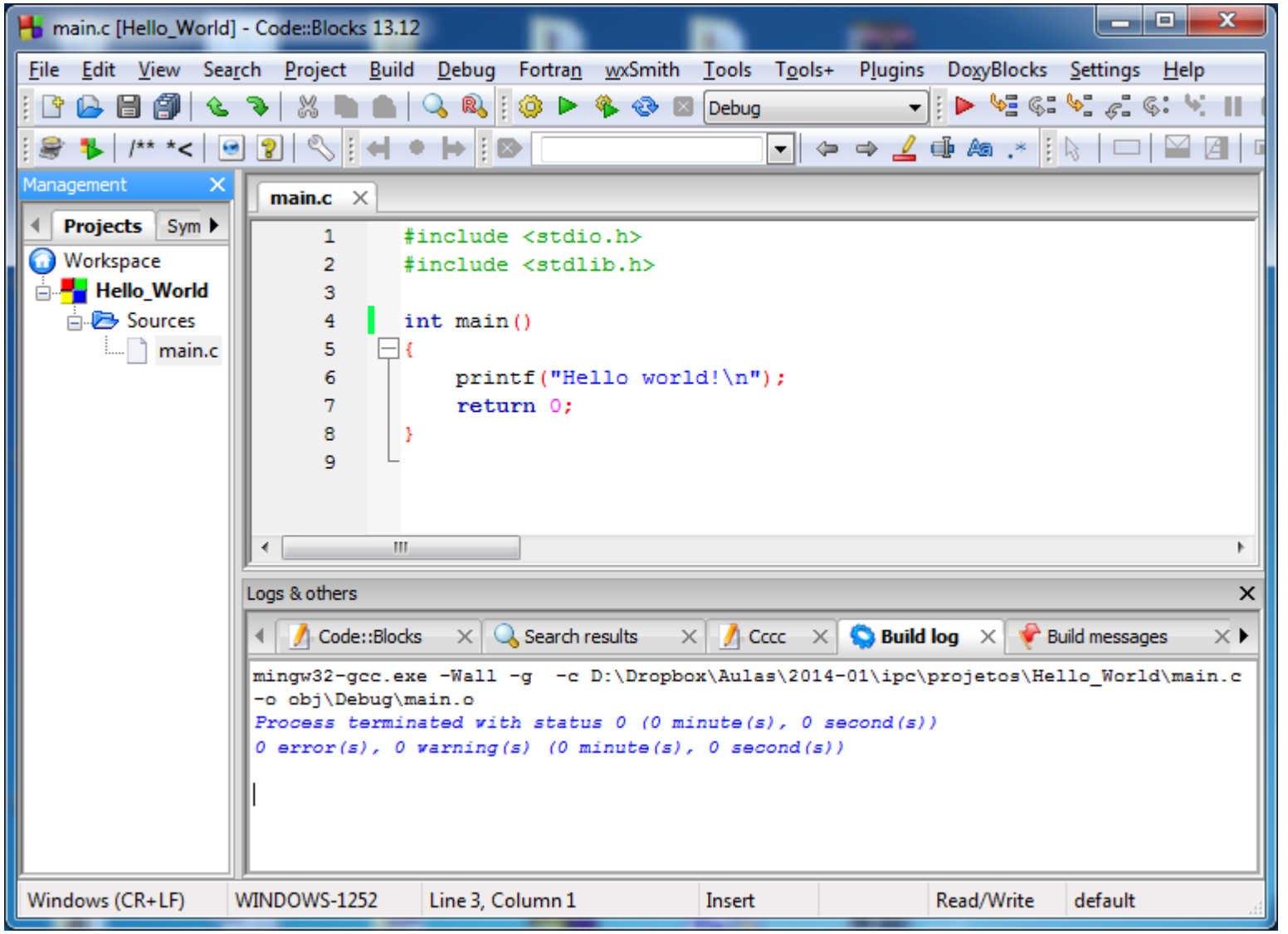
- Conjunto de programas que transforma uma linguagem de alto nível em linguagem de máquina executável por computador

```
#include<stdio.h>
void torre(int n, char source, char dest,char aux);

main()
{
      int n;
      char source = 'A';
      char dest = 'C';
      char aux = 'B';
      printf("entre com o numero de discos: ");
      scanf("%d",&n);
      torre(n, source,dest,aux);
      getchar();
      getchar();
}

void torre(int n, char source, char dest,char aux)
{
     static int step = 0;

     printf("torre(%d, %c, %c, %c)\n",n, source, dest,aux);
     if (n==1)
       printf("\t\t\tstep %3d: mova de %c para %c\n",++step, source, dest);
     else
```



```
0001000010001110001011100110100100101001010011111
0101010100101010
```

Após Ctrl_Shift+F9 (Compile)



Compilando o código fonte

mingw32-gcc.exe é um programa que realiza a compilação

---------- Build: Debug in Hello_World (compiler: GNU GCC Compiler)----------

mingw32-gcc.exe -Wall -g -c D:\Dropbox\Aulas\2014-01\ipc\projetos\Hello_World\main.c -o obj\Debug\main.o

Process terminated with status 0 (0 minute(s), 0 second(s))

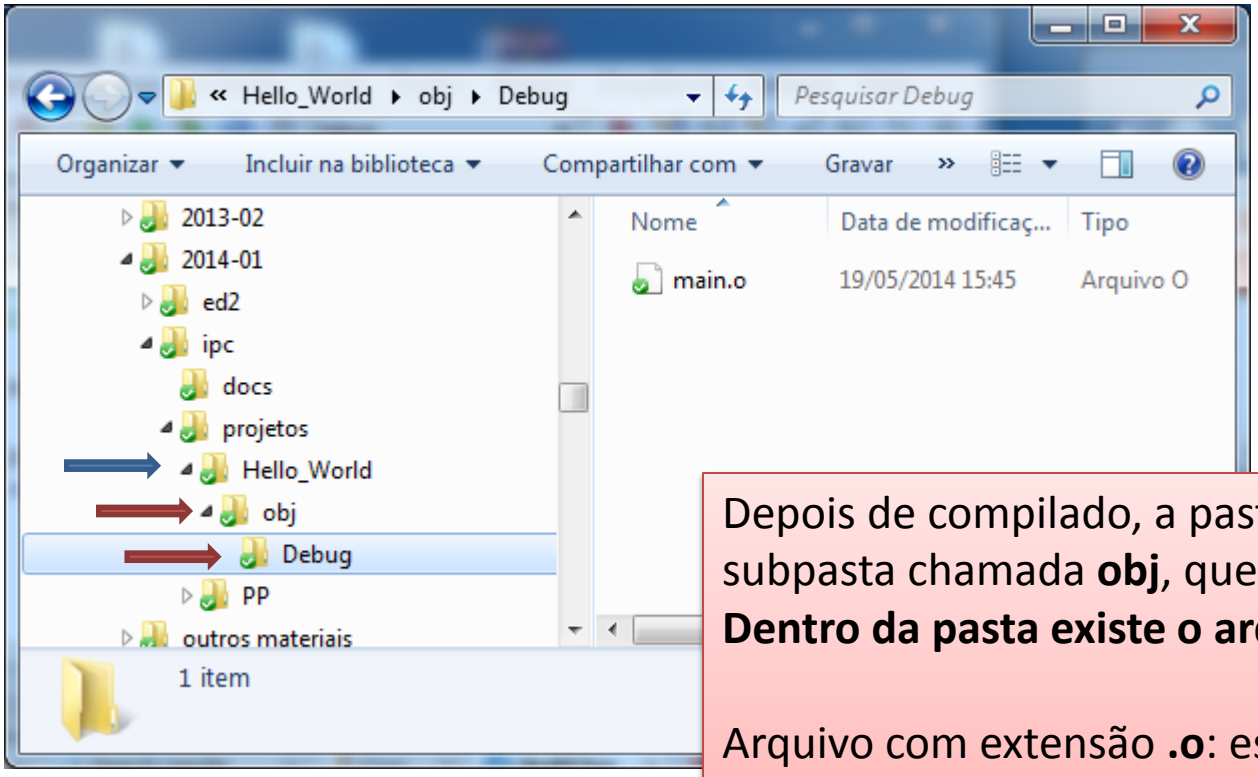
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

Vale observar que Code::Blocks **não é o compilador**: ele é somente uma interface gráfica (um ambiente de desenvolvimento) que **chama o compilador**

main.o: Este é o código com o resultado da compilação – o arquivo objeto (.o)

main.c: Este é código fonte do programa que escrevemos

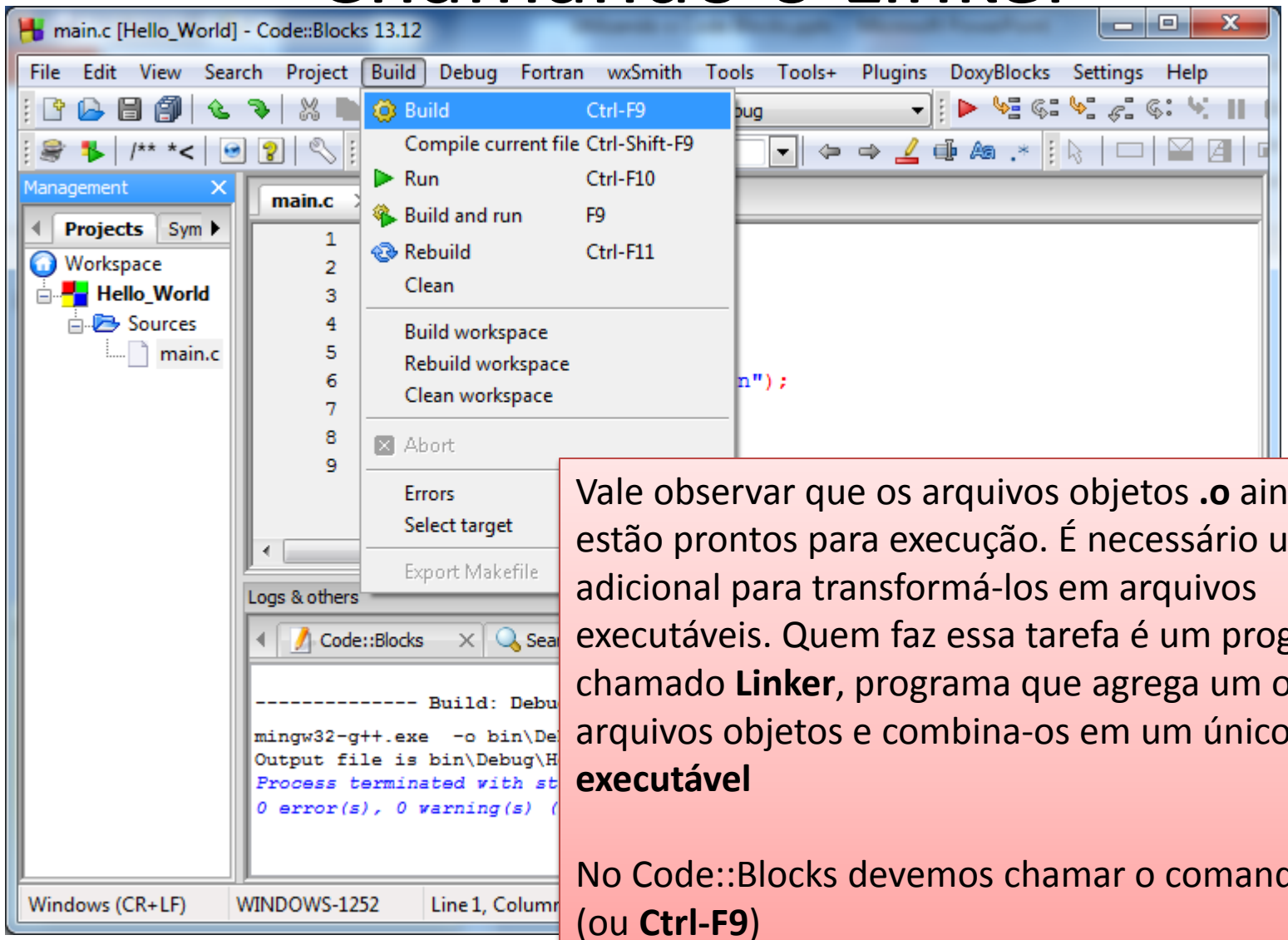
Após a compilação



Depois de compilado, a pasta agora contém uma subpasta chamada **obj**, que contém a subpasta **Debug Dentro da pasta existe o arquivo chamado main.o**

Arquivo com extensão **.o**: esse é o arquivo em código de máquina, compilado. Ele é chamado de **arquivo objeto.** Ele é a tradução do arquivo .c para código de máquina

Chamando o Linker



Vale observar que os arquivos objetos **.o** ainda não estão prontos para execução. É necessário um passo adicional para transformá-los em arquivos executáveis. Quem faz essa tarefa é um programa chamado **Linker**, programa que agrega um ou mais arquivos objetos e combina-os em um único arquivo **executável**

No Code::Blocks devemos chamar o comando **Build** (ou **Ctrl-F9**)

Compilando o código fonte

mingw32-g++.exe é um programa que realiza o processo de 'linker'

Vale observar que Code::Blocks **não é o compilador**: ele é somente uma interface gráfica (um ambiente de desenvolvimento) que **chama o linker**

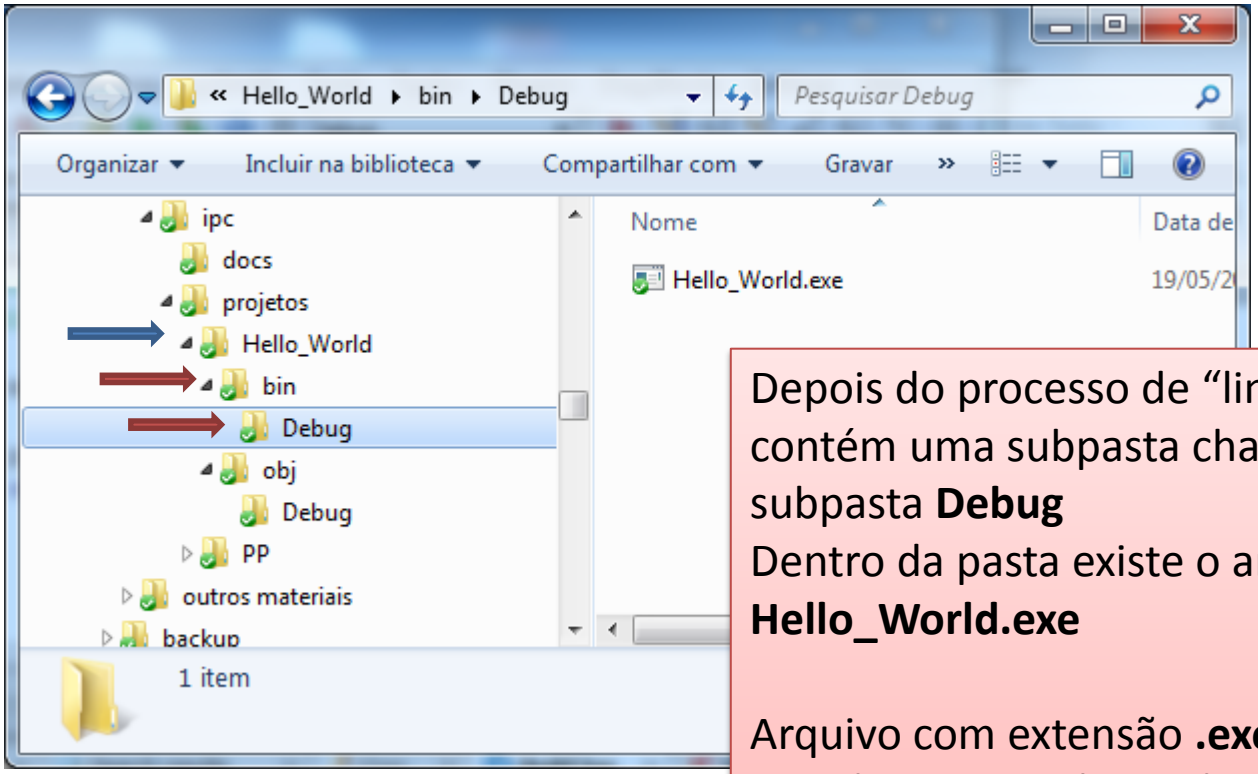
---------- **Build: Debug in Hello_World (compiler: GNU GCC Compiler)**----------

mingw32-g++.exe -o bin\Debug\Hello_World.exe obj\Debug**main.o**
Output file is bin\Debug**Hello_World.exe** with size 27.96 KB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

Hello_World.exe: Este é o código com o resultado da "linkagem".

main.o: Este é código objeto a ser juntado ("linkado") a outros códigos pré-existentes (chamados bibliotecas)

Após a compilação

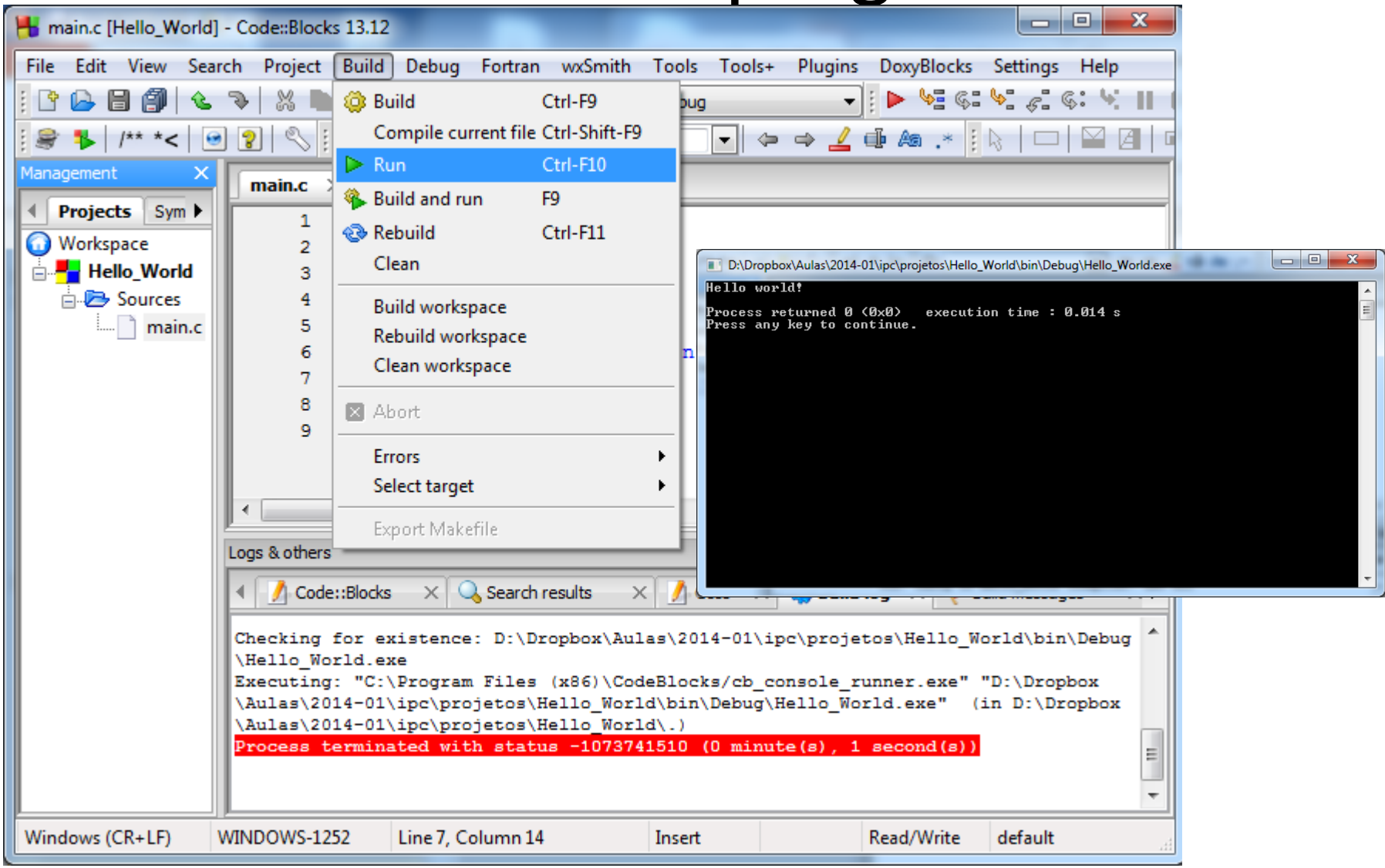


Depois do processo de “linkagem”, a pasta agora contém uma subpasta chamada **bin**, que contém a subpasta **Debug**
Dentro da pasta existe o arquivo chamado **Hello_World.exe**

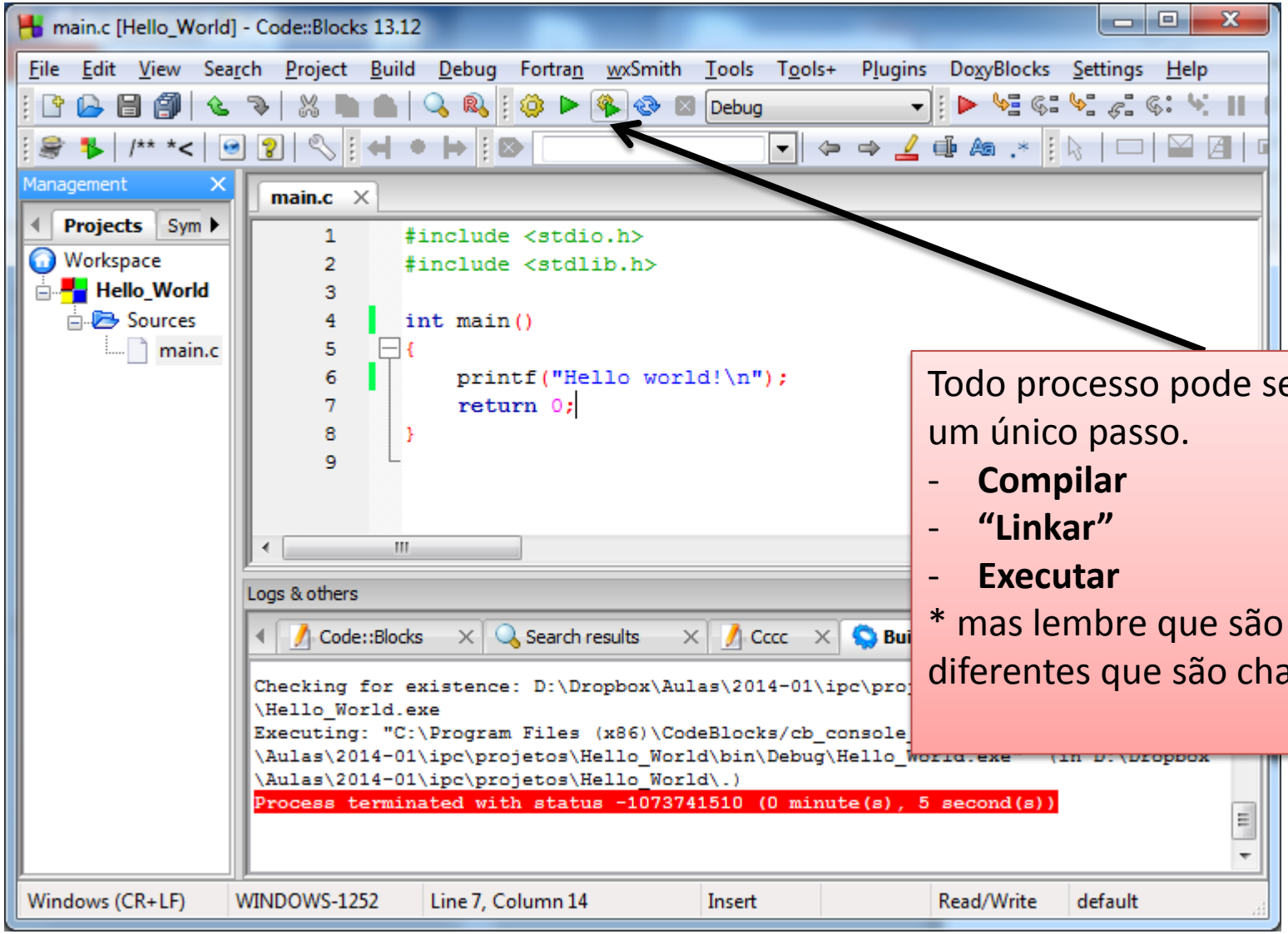
Arquivo com extensão **.exe**: no Windows, esse é um tipo de arquivo chamado executável, pois você consegue executá-lo (você pode com o duplo clique do mouse abrir o programa que acabou de ser criado)

No Linux não há necessidade da extensão ser .exe, e sim que o arquivo tenha a permissão ‘executável’

Executando o programa



Build and Run (tecla F9)



Todo processo pode ser feito em um único passo.

- **Compilar**
- **“Linkar”**
- **Executar**

* mas lembre que são programa diferentes que são chamados

Prática

- (1) Faça um programa que realize a seguinte operação
- Y = A * X + 2 * B + C

Prática

- Calcule o IMC

$$IMC = \frac{massa\ (kg)}{altura(m)^2}$$

IMC	Classificação
< 18,5	Magreza
18,5 – 24,9	Saudável
25,0 – 29,9	Sobrepeso
30,0 – 34,9	Obesidade Grau I
35,0 – 39,9	Obesidade Grau II (severa)
≥ 40,0	Obesidade Grau III (morbida)

(fonte: Wikipedia)

Referências

- Slides adaptadas da aula da Prof. Denise Guliato Faculdade de Computação