

Introdução aos Computadores e às Linguagens de Programação

Disciplina de Programação de Computadores I
Universidade Federal de Ouro Preto

Agenda

- O que é um computador?
- Como um computador é organizado internamente?
- Como funciona um computador?
- O que são algoritmos?
- O que são Linguagens de Programação?
- Como são criados e executados os programas?



O que é um computador?

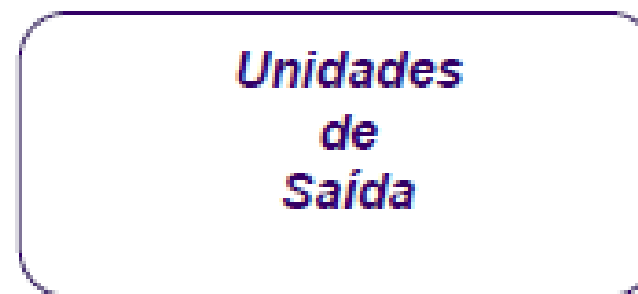
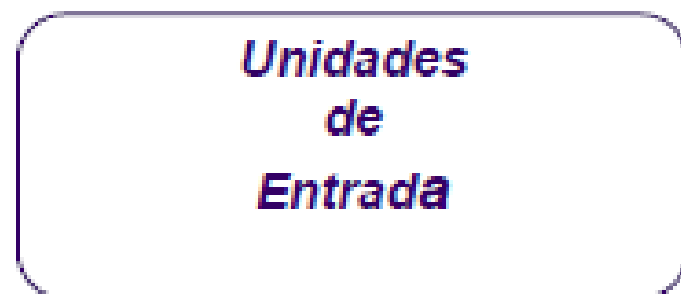
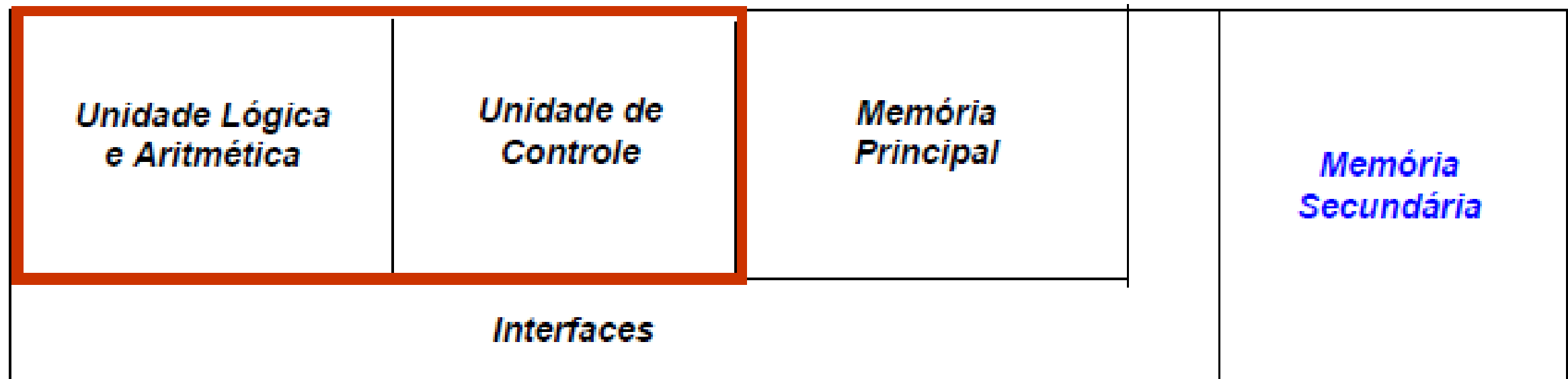
- É um dispositivo capaz de realizar computações (cálculos) e tomar decisões lógicas muito mais rapidamente que nós humanos.
- É composto de parte física (*hardware*) e parte lógica (*software*).
- Ele processa informações de acordo com um conjunto de comandos que formam um programa.
- Os comandos são escritos em uma linguagem de programação.

Organização do computador

- O hardware do computador é entendido pelo software em 6 partes:
 - Unidades de entrada de dados: teclado, mouse, etc.;
 - Unidades de saída de dados: monitor, impressora, rede, etc.;
 - Unidade de memória principal: memória RAM;
 - Unidade lógica e aritmética (ULA);
 - Unidade central de processamento (CPU);
 - Unidade de armazenamento secundário: Harddisk, DVD, pendrive.

Modelo lógico do computador

Unidade Central de Processamento



Como funciona o computador?

- Um programa é uma sequência de 0s e 1s armazenado na memória do computador.
- O programa é executado pela CPU, que interpreta as sequências de 0s e 1s como comandos.
- Aritmética binária inteira: cálculos são feitos com números inteiros representados na base 2.
- Bit: Menor unidade de informação (representa os dois estados - 0 e 1- da lógica binária).

Como funciona o computador?

- Conceitos: **Bit**, **Byte** e **Word**
 - **Bit** (“**B**inary **DigiT**” – dígito binário)
 - Unidade de Informação, tem somente os valores “0” ou “1”.
 - **Byte** (“**B**inar**Y** **T**erm” – termo binário)
 - Conjunto de 8 bits, com o qual pode-se representar os números, as letras, os sinais de pontuação, etc.
 - **Palavra (Word)**
 - É a quantidade de bits que a CPU processa por vez.

Representação binária

- O número 5 (base 10) equivale a 101 (base 2).
- O número 101 (base 2) pode ser representado como:
 - 101 (base 2): mínimo de bits necessário
 - 0101 (base 2): com 4 bits
 - 0000 0000 0000 0101 (base 2): com 16 bits
 - 0000 0000 0000 0000 0000 0000 0000 0101 (base 2): com 32 bits

Representação binária

- Conceitos: **Bit**, **Byte** e **Word**

• CPU ou micro de:	Palavra de:
8 bits	8 bits = 1 byte = 1 caractere
16 bits	16 bits = 2 bytes = 2 caracteres
32 bits	32 bits = 4 bytes = 4 caracteres
64 bits	64 bits = 8 bytes = 8 caracteres
128 bits	128 bits = 16 bytes = 16 caracteres

- Exemplo: se a palavra (texto) **TRADUTOR** tiver sido transferida da memória para uma CPU de:
 - 8 bits <= este precisará de 8 operações para processá-la;
 - 16 bits <= este precisará de 4 operações para processá-la;
 - 32 bits <= este precisará de 2 operações para processá-la;
 - 64 bits <= este precisará de uma operação para processá-la;

Sistemas Computacionais

- **Unidades de Medida:**

Unidades	Usual	Informática
Kilo (K)	10^3	2^{10} bytes
Mega (M)	10^6	2^{20} bytes
Giga (G)	10^9	2^{30} bytes
Tera (T)	10^{12}	2^{40} bytes

Sistemas Computacionais

- **Memória Principal**

- Representação de uma memória de 1 KB:

- O processador acessa o conteúdo de um byte a partir do endereço desse byte.

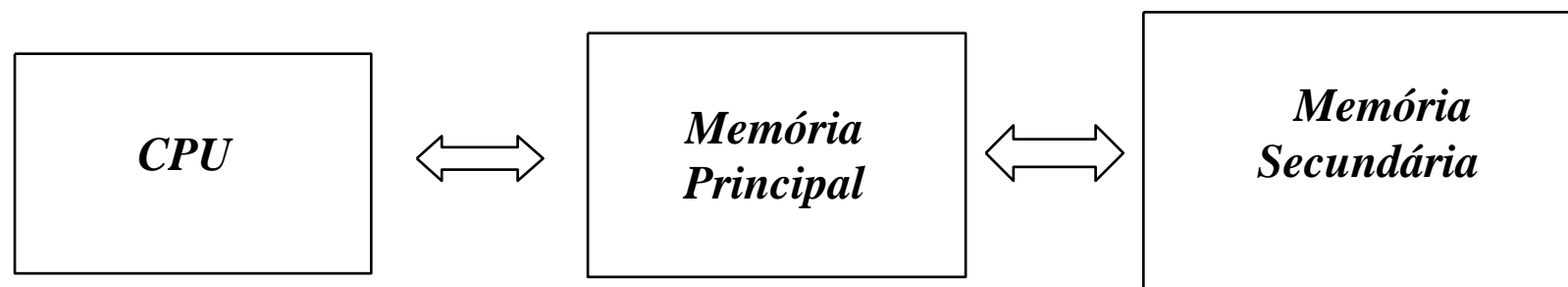
Endereço	Byte							
0								
1								
2	0	1	0	0	0	0	0	1
...								
1023								

No byte de endereço 2 está armazenado o código ASCII do caractere "A".

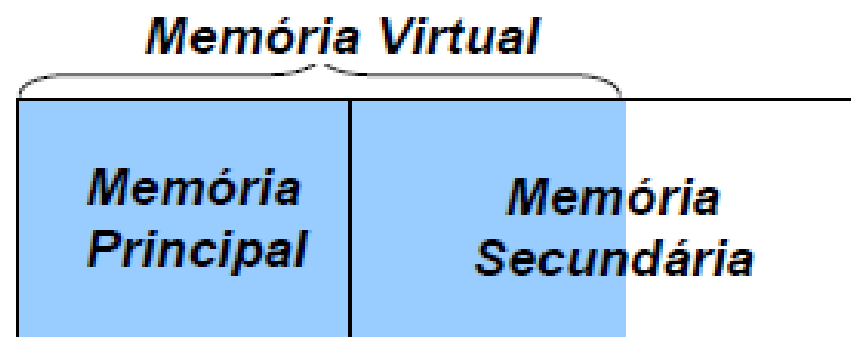
Sistemas Computacionais

- **Memória Secundária**

- A memória secundária pode ser composta por vários dispositivos capazes de ampliar a capacidade de armazenamento da memória principal. Eles podem armazenar grandes quantidades de dados e programas.
- A memória secundária é um tipo de memória não volátil, teoricamente permanente e mais lenta.



- Outra função da memória secundária é oferecer uma expansão virtual da memória principal – **Memória Virtual**.



Algoritmos

- São sequências de passos, precisos e bem definidos, que descrevem como realizar uma tarefa
- Podem ser especificados em português, português estruturado, fluxogramas, linguagens de programação, etc.

Algoritmo em português

Calcule a soma dos números 1234 e 456.

Escrever os números em um papel, um abaixo do outro, alinhados pelo dígito das unidades. Para cada coluna de 1 ou mais dígitos, somar os dígitos alinhados. Caso o valor da soma ultrapasse 9, adicionar 1 à coluna imediatamente à esquerda da coluna atual e anotar o dígito das unidades desta soma como o resultado da soma. Após executar todas as somas, ler o resultado final.

Algoritmo em Português Estruturado

Escreva os números em um papel, um abaixo do outro, alinhados pelo dígito das unidades.

Para cada coluna de 1 ou mais dígitos:

- Some os dígitos alinhados.

- Caso o valor da soma ultrapasse 9:

- Adicione 1 à coluna imediatamente à esquerda da coluna atual.

- Anote o dígito das unidades desta soma como o resultado da soma.

- Use a soma criada como resposta.

Exercício

- Dados N números, escrever um algoritmo em português estruturado que calcule a média destes números.

Algoritmos

- Como estudar algoritmos?
 - O aprendizado de algoritmos não se consegue a não ser através de muitos exercícios.
 - Algoritmos **não se aprendem**:
 - Copiando algoritmos
 - Estudando algoritmos
 - Algoritmos **só se aprendem**:
 - Construindo algoritmos
 - Testando algoritmos

Linguagens de Programação

- São o mecanismo que utilizamos para dar ordens para os computadores.
- Podem ser de 3 tipos:
 - linguagem de máquina: diretamente entendidas pelos computadores;
 - linguagens de montagem: precisam de um programa montador para gerar linguagem de máquina
 - linguagens de alto nível: precisam de um compilador para traduzi-las para uma linguagem de montagem.

Linguagens de Máquina

- É uma sequência de 0s e 1s que pode ser executada por um computador.
- Cada processador tem uma linguagem de máquina própria, ou seja, entende sequências de 0s e 1s como comandos diferentes.

Linguagem de Montagem

- Linguagem que utiliza siglas para representar comandos em sequências de 0s e 1s. Ex: Assembly.
- Um montador traduz este código para linguagem de máquina.

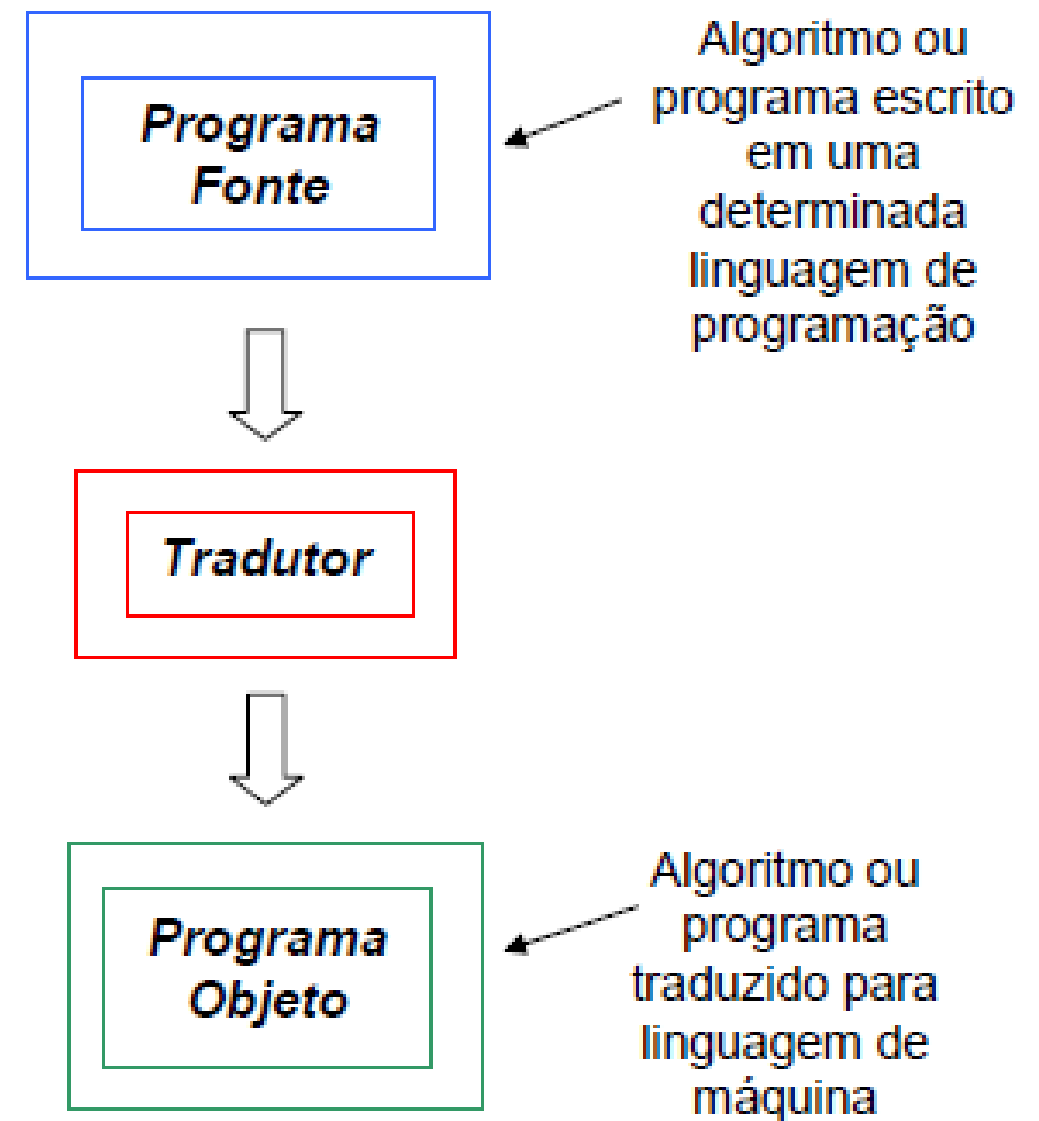
```
1  TITLE Hello                (...this is a comment area...good to give name of program)
2
3  ; This program puts basic output on the screen
4  ; Last update: 8/9/02
5
6  INCLUDE Irvine32.inc       ;libraries being called put here
7
8  .data
9  str1 byte "hello",0        ;word hello stored in the variable of type byte called str1
10 .code
11 main PROC ;program begins
12     mov edx, offset str1    ;message moved into a register where strings can be held
13     call WriteString        ;executes display of the string
14     call Crlf               ;gives a line feed
15     exit                   ;program exits here
16 main ENDP                  ;end of main function
17 END main
```

Linguagens de Programação (de Alto Nível)

- São linguagens cujos comandos são mais próximos da linguagem humana.
- Ex: C, C++, Java, Haskell, Prolog, etc.
- O compilador é responsável por ler um código nesta linguagem e traduzi-lo para uma linguagem de montagem. Ex: GCC, Visual Studio, etc.
- O montador, por sua vez, gera um código executável em um determinado computador.

Linguagens de Programação (de Alto Nível)

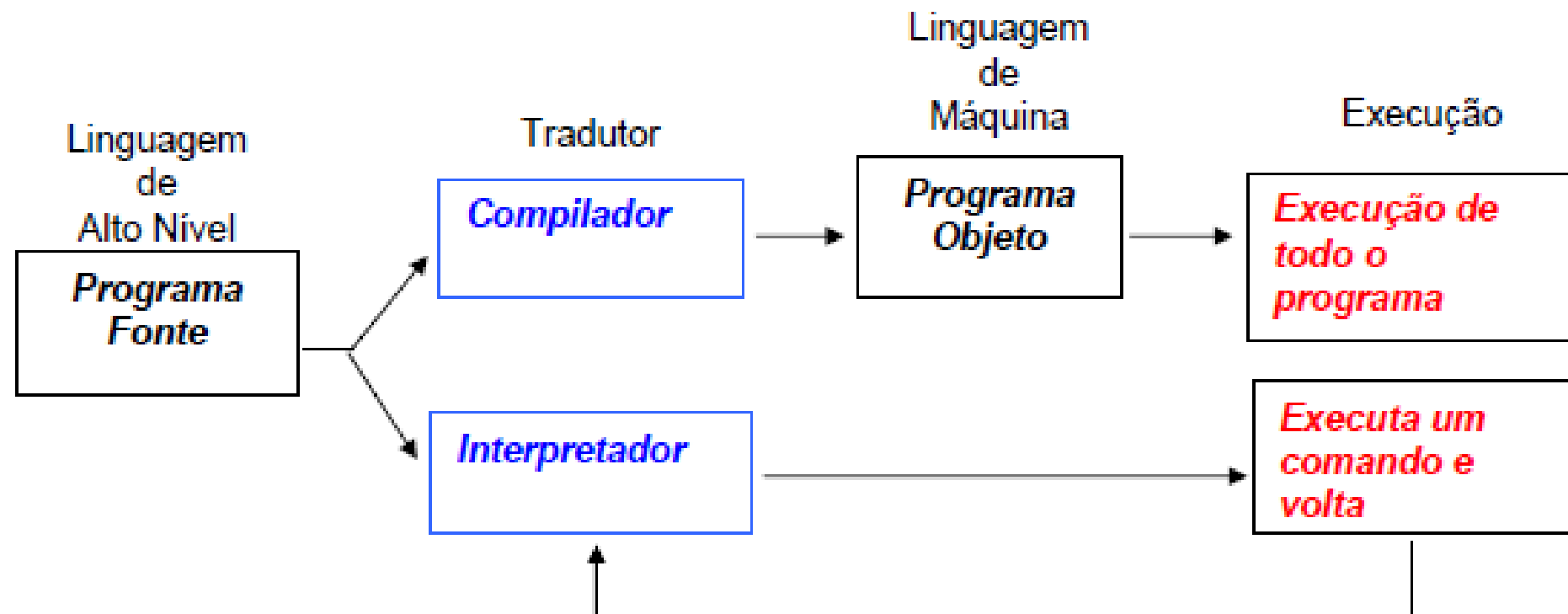
- Os computadores só podem executar **diretamente** os algoritmos expressos em linguagem de máquina (que é um conjunto de instruções capazes de ativar diretamente os dispositivos eletrônicos do computador).
- Um tradutor é um programa que traduz um algoritmo que está escrito em uma determinada linguagem de programação em linguagem de máquina.



Linguagens de Programação (de Alto Nível)

- **Processo de Tradução:**

- O Processo de tradução pode ser feito por:
 - Compilação: Lê, analisa e traduz todos os comandos do programa fonte, criando o programa objeto.
 - Interpretação: Traduz ou interpreta cada comando ao executá-lo.



Do algoritmo para um programa em C

1. O programador escreve o algoritmo na linguagem C em um arquivo de texto, chamado de código fonte.
2. O compilador gera códigos objeto a partir da compilação dos códigos fonte.
3. Os códigos objeto de um ou mais códigos fonte são unidos pelo *linker*, gerando um programa executável.
4. O código executável é carregado na memória pelo *loader*.
5. A CPU executa as instruções carregadas.
6. Quando encontramos erros no funcionamento do programa, podemos depurá-lo com um depurador.

Escrevendo um programa em C

Suponha o arquivo *fonte.c* com o conteúdo a seguir:

```
#include <stdio.h>
```

```
// Isto é um comentário!
```

```
int main( void ) {
```

```
    int resultado = 1234+456;
```

```
    printf("1234 + 456 = %d \n" , resultado);
```

```
    return 0;
```

```
}
```

Estrutura de um programa em C

Declaração de Bibliotecas Utilizadas

// Comentários

/* Comentários */

int main(void){

 Declaração de Variáveis

 Comandos

}

Identificando e corrigindo erros: Depuração

- Os programas podem possuir erros:
 - que os impeçam de ser compilados;
 - na implementação do algoritmo ou em sua lógica, produzindo respostas erradas.
- Para identificar o segundo tipo de erros utiliza-se um depurador de código.
- O depurador permite executar o programa passo a passo, inspecionando a memória durante a execução.

Algoritmos

- **IDE - Integrated Development Environment**
 - Ambiente Integrado de Desenvolvimento
 - Programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.
 - Editor, compilador, depurador, etc.

Algoritmos

- **IDE - CodeBlocks**

- CodeBlocks: IDE disponível para Linux, Mac e Windows
 - <http://www.codeblocks.org/downloads>
- Download gratuito

Algoritmos

- **IDE - CodeBlocks**



Code::Blocks Code::Blocks - The IDE with all the features

Home Features Downloads Forums Wiki

Main

- Home
- Features
- Downloads
 - Binaries
 - Source
 - SVN
- Plugins
- User manual
- Licensing
- Donations

Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- BugTracker
- PatchTracker
- Browse SVN
- Browse SVN log

Downloads

There are different ways to download and install Code::Blocks on your computer:

- **Download the binary release**

This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer.
- **Download a nightly build:** There are also more recent so-called *nightly builds* available by the community (Big "Thank you" for that!) Please note that we consider nightly builds to be unstable.
- **Download the source code**

If you feel comfortable building applications from source, then this is the recommended way to get the latest versions or, even better, create patches for bugs you may find and contributing them back to the community.
- **Retrieve source code from SVN**

This option is the most flexible of all but requires a little bit more work to setup. It gives you the latest source code and allows you to track changes and contribute bug-fixes!


Besides Code::Blocks itself, you can compile extra plugins from contributors to extend its functionality.

Thank you for your interest in downloading Code::Blocks!



Algoritmos

• IDE - CodeBlocks



Code::Blocks

Code::Blocks - The IDE with all the features you need, having a consistent look, feel and operation across platforms.


- Home
- Features
- Downloads
- Forums
- Wiki


Main


- Home
- Features
- Screenshots
- Downloads
 - Binaries
 - Source
 - SVN
- Plugins
- User manual
- Licensing
- Donations


Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- BugTracker
- PatchTracker
- Browse SVN
- Browse SVN log



built with 





Please select a setup package depending on your platform:

- Windows 2000/XP/Vista/7/8
- Linux 32-bit
- Linux 64-bit
- Mac OS X

NOTE: There are also more recent *nightly builds* available in the [forums](#) or (for Debian and Fedora users) in [Jens' Debian repository](#) and [Jens' Fedora repository](#). Please note that we consider nightly builds to be *stable*, usually.

MIRRORS: BerliOS mirrors all files *usually* at SourceForge using a "BerliOS robot" [here](#). As this sometimes doesn't work, we have mirrored all file releases at SourceForge, too [here](#). The latter is managed by us.

IMPORTANT NOTE: If you try to download from BerliOS and get a "Too many clients!" - error, you should retry to download the file. According to a BerliOS - admin, this can happen several times, before the download starts. Alternatively use on of the mirrors.

NOTE: We have a [Changelog for 12.11](#), that gives you an overview over the enhancements and fixes we have put in the new release.

Windows 2000 / XP / Vista / 7:

File	Date	Size	Download from
codeblocks-12.11-setup.exe	28 Nov 2012	28.2 MB	BerliOS or Sourceforge.net
codeblocks-12.11-setup_user.exe			BerliOS or Sourceforge.net
codeblocks-12.11mingw-setup.exe	28 Nov 2012	96.8 MB	BerliOS or Sourceforge.net
codeblocks-12.11mingw-setup_user.exe			BerliOS or Sourceforge.net

NOTE: The codeblocks-12.11mingw-setup.exe file *includes* the GCC compiler and GDB debugger from TDM-GCC (version 4.7.1, 32 bit).

NOTE: The codeblocks-12.11(mingw)-setup_user.exe will NOT request ADMIN rights and can be installed into write accessible folders only. Trying to install to a folder like "Program Files" will result in an access error therefore. Use this special installer if you do not have admin access on your Windows machine. IF UNSURE, USE "codeblocks-12.11mingw-setup.exe"!

Linux 32-bit:

Distro	File	Date	Size	Download from
--------	------	------	------	---------------

Algoritmos

- **IDE - CodeBlocks**



Windows XP / Vista / 7 / 8.x / 10:

File

codeblocks-16.01-setup.exe

codeblocks-16.01-setup-nonadmin.exe

codeblocks-16.01-nosetup.zip

codeblocks-16.01mingw-setup.exe

codeblocks-16.01mingw-nosetup.zip

codeblocks-16.01mingw_fortran-setup.exe



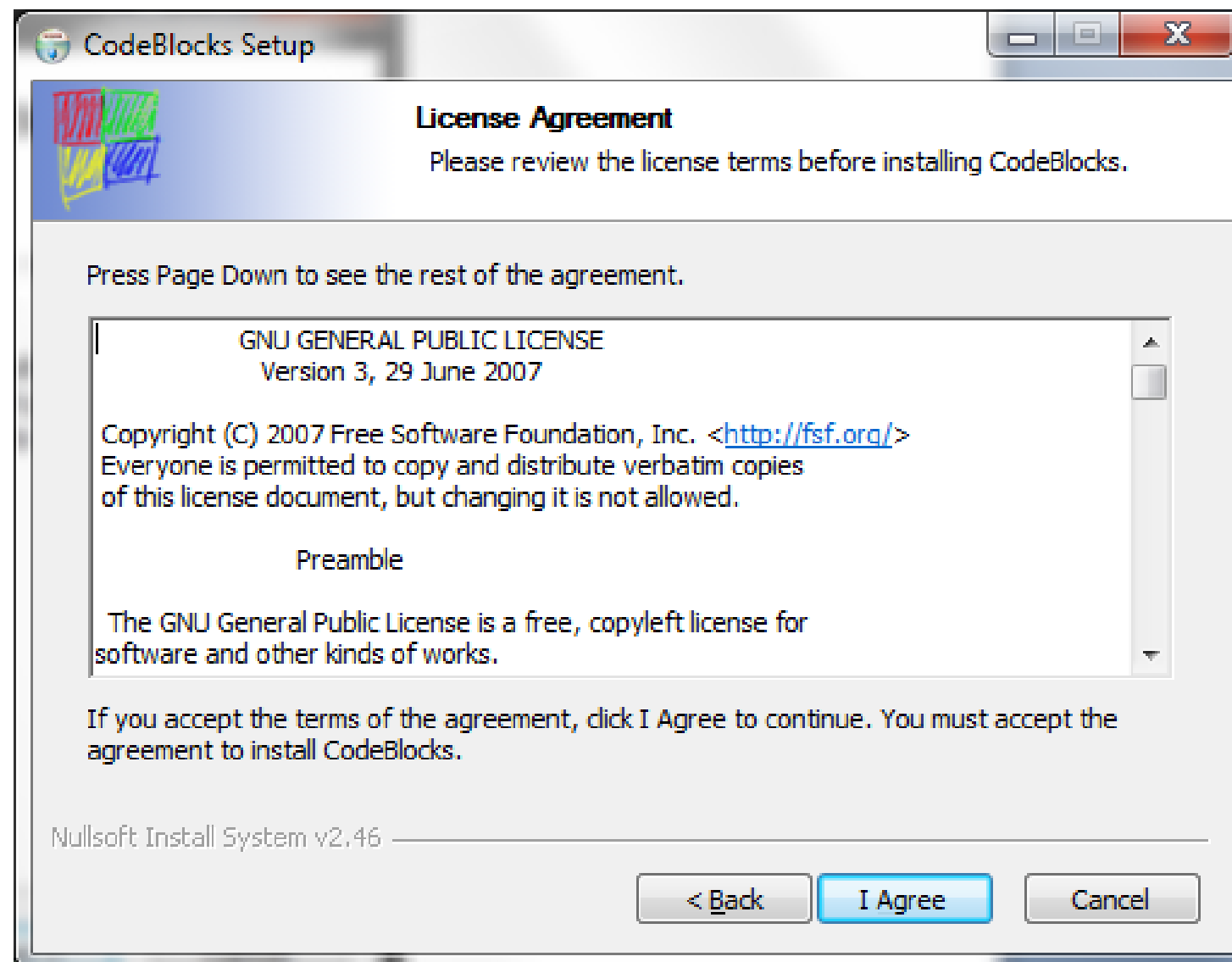
Algoritmos

- **IDE - CodeBlocks**



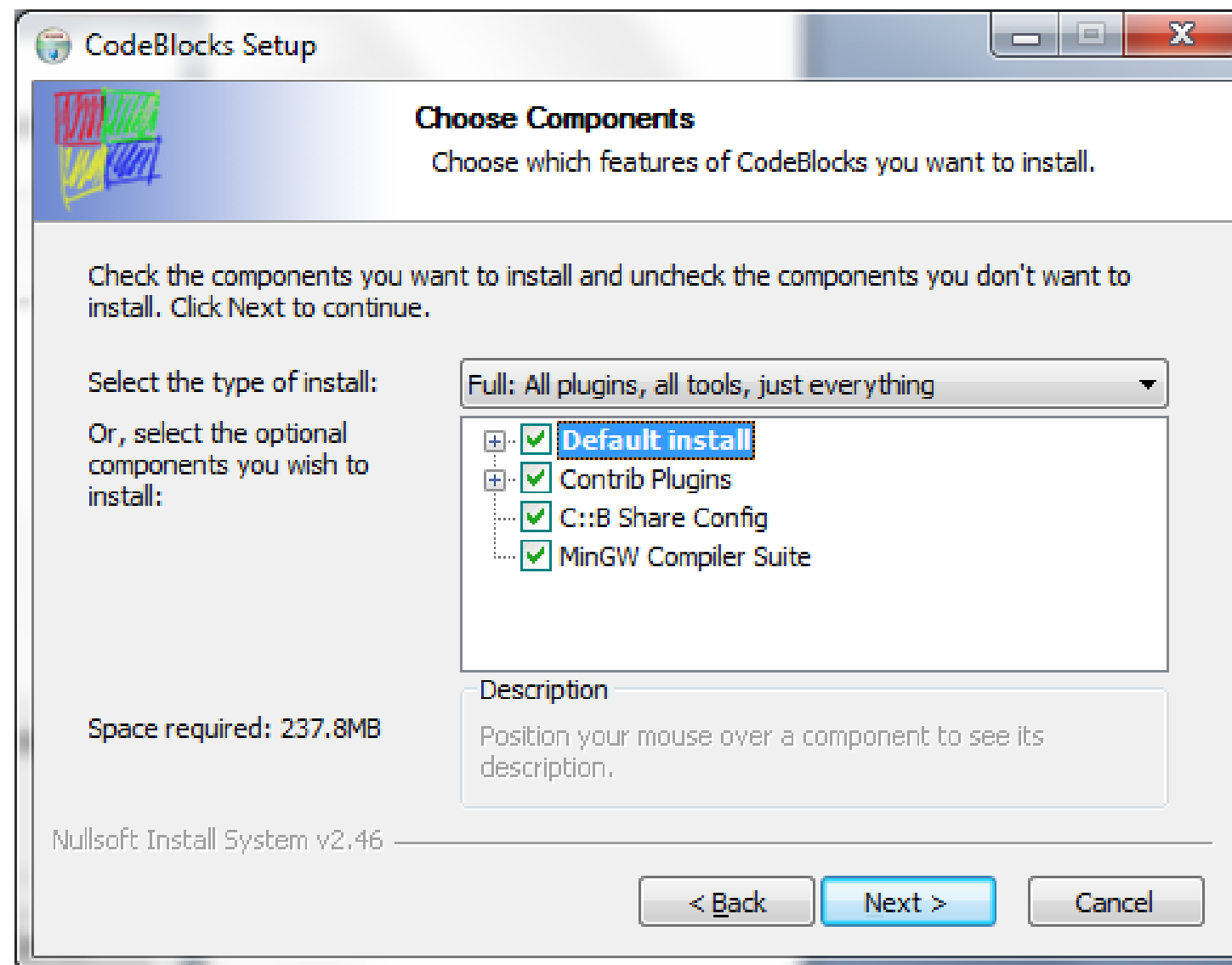
Algoritmos

- **IDE - CodeBlocks**



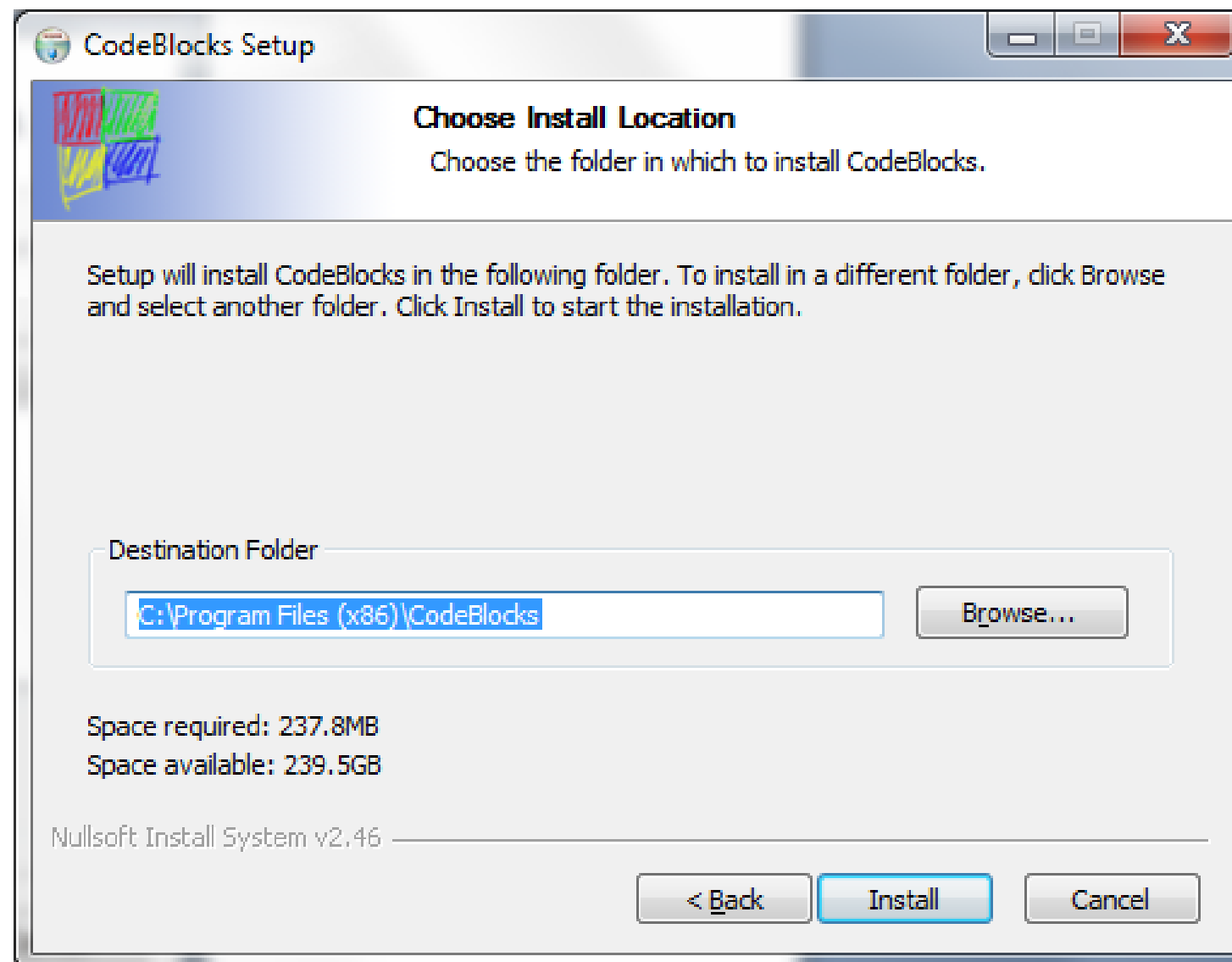
Algoritmos

- **IDE - CodeBlocks**



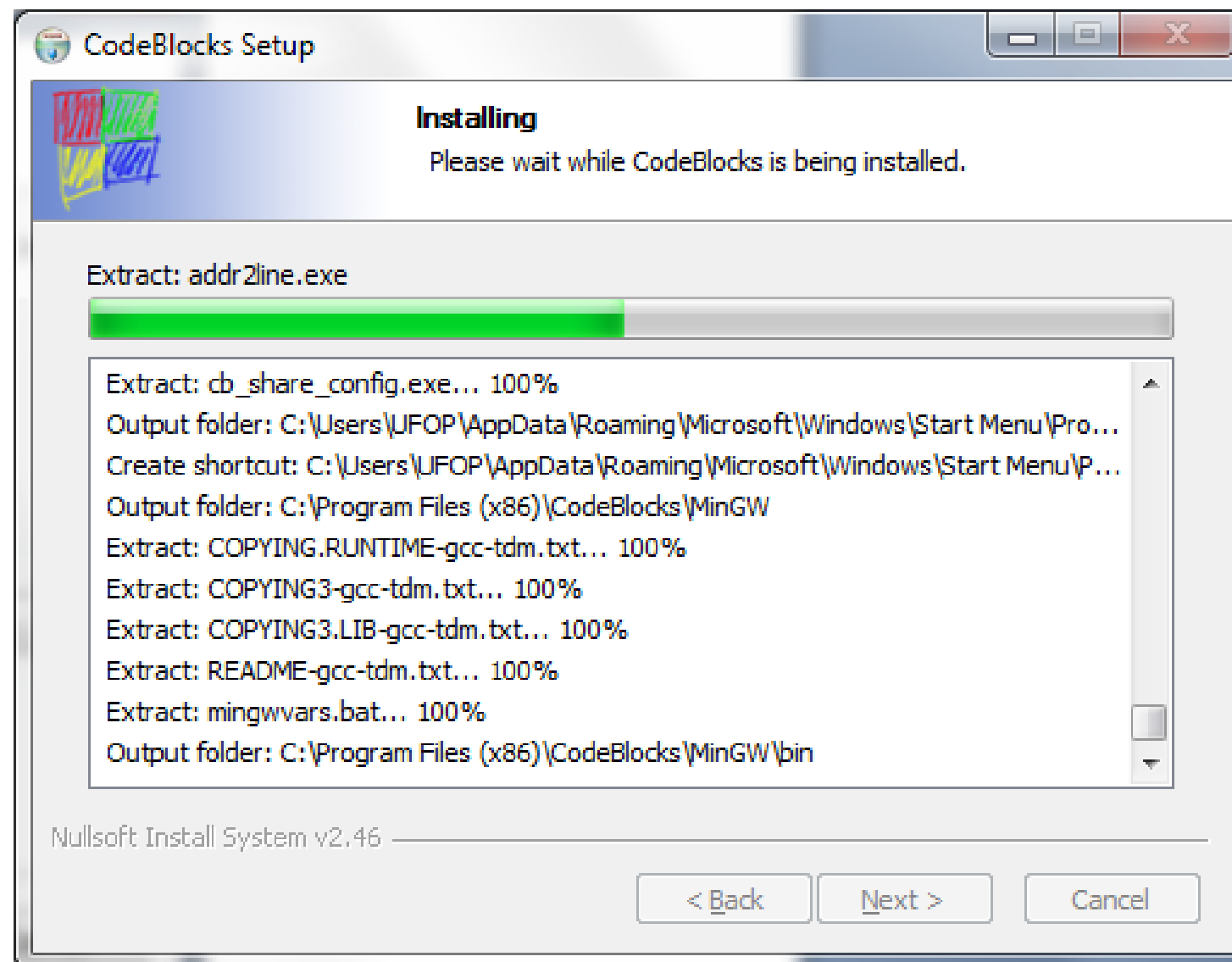
Algoritmos

- **IDE - CodeBlocks**



Algoritmos

- **IDE - CodeBlocks**



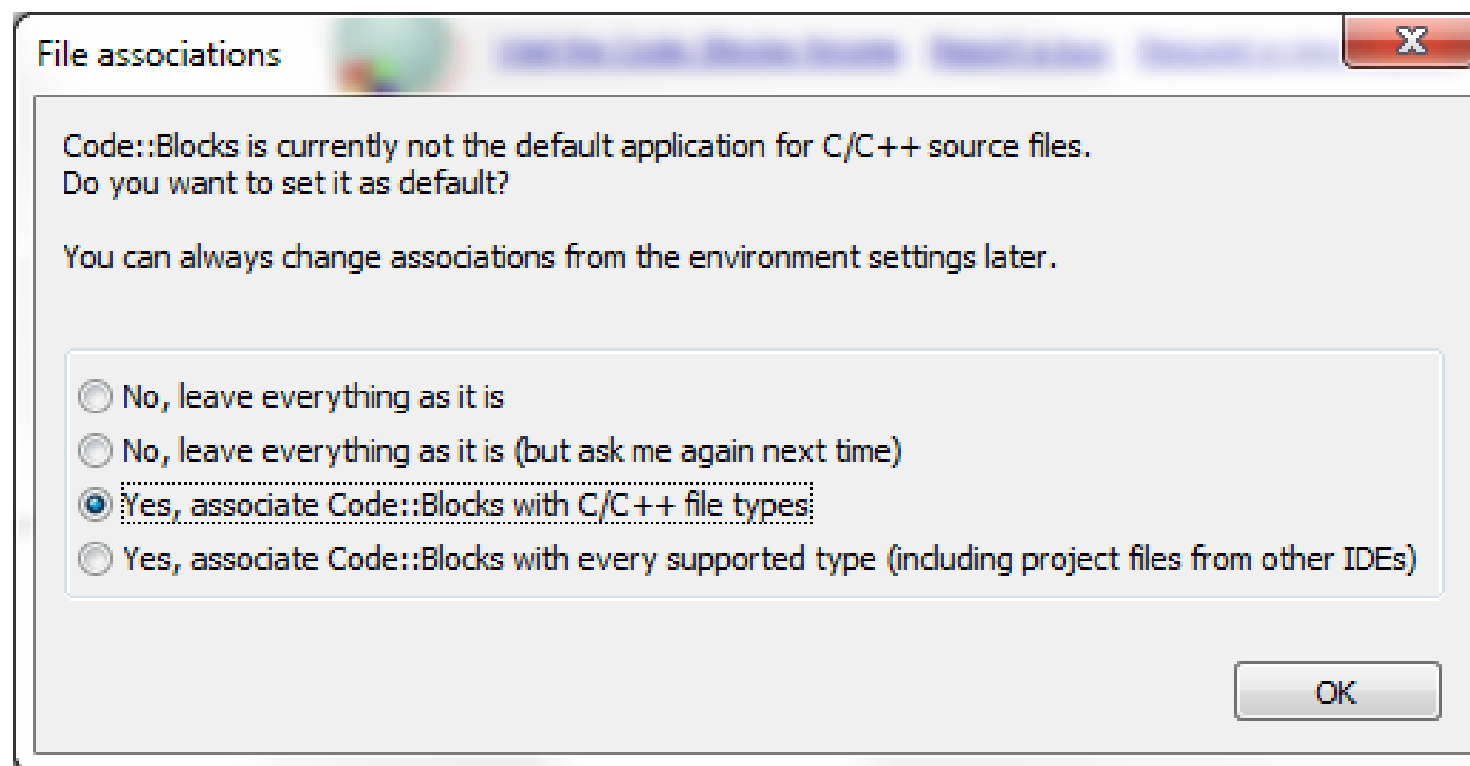
Algoritmos

- **IDE - CodeBlocks**



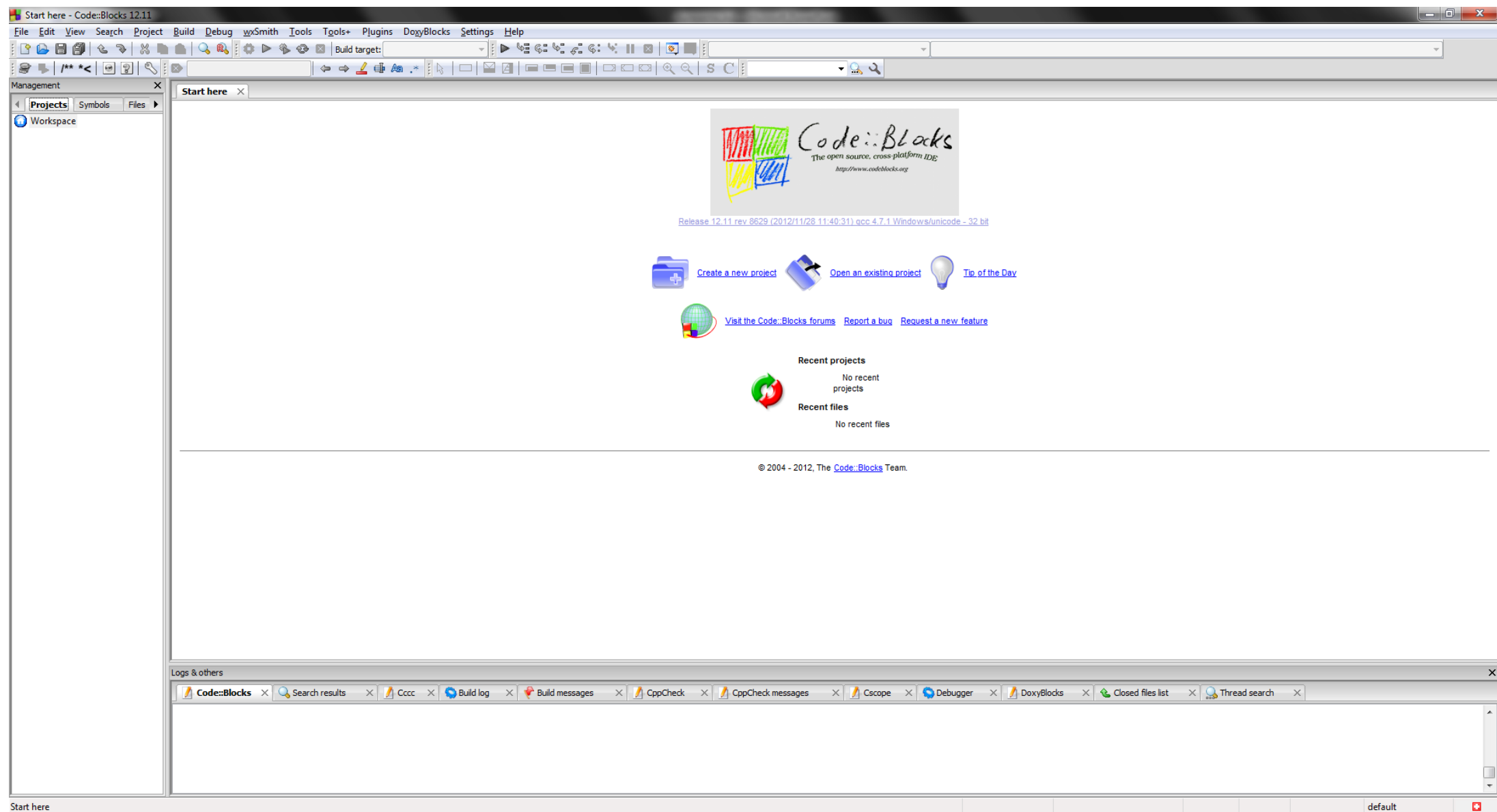
Algoritmos

- **IDE - CodeBlocks**



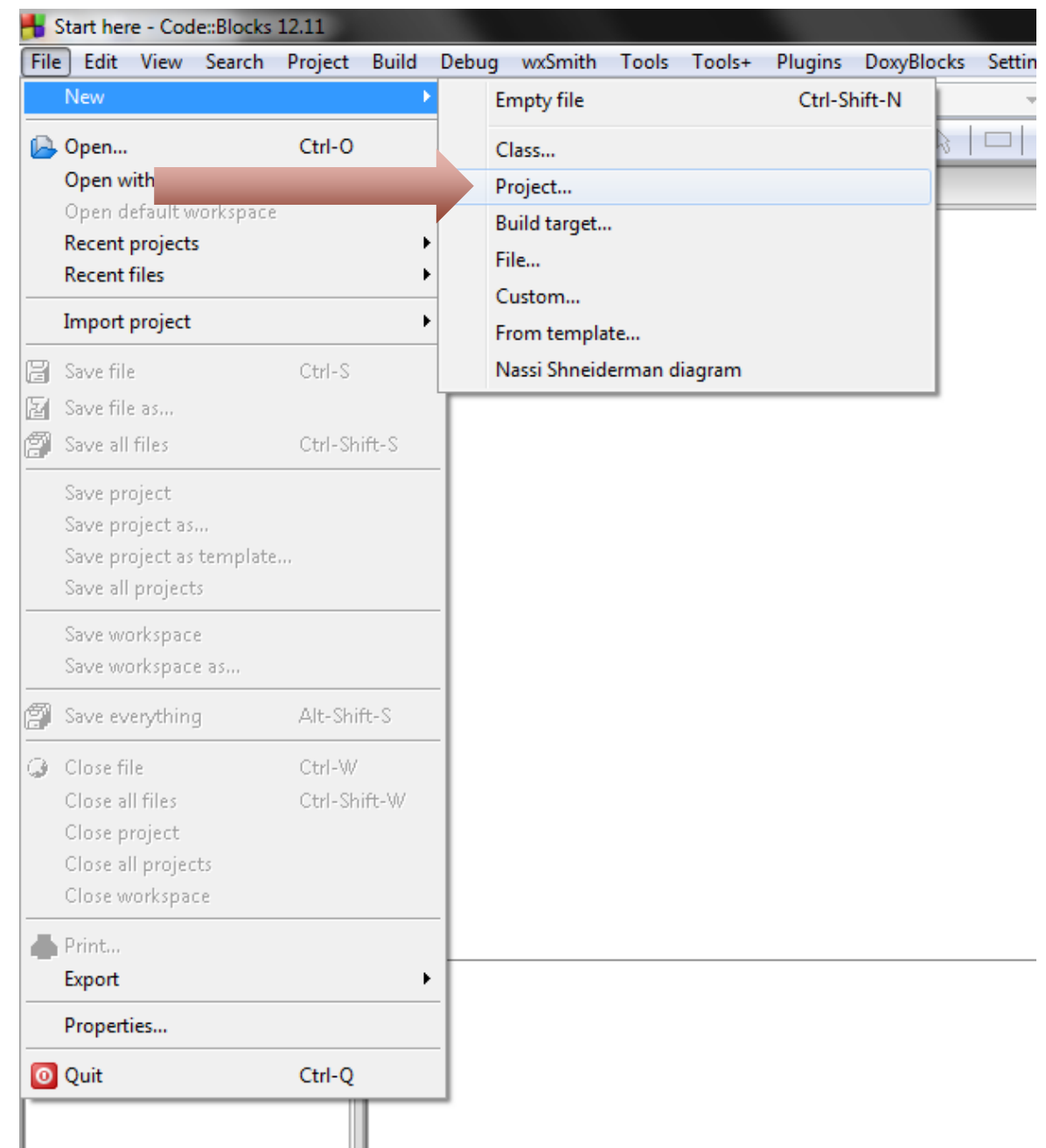
Algoritmos

- **IDE - CodeBlocks**



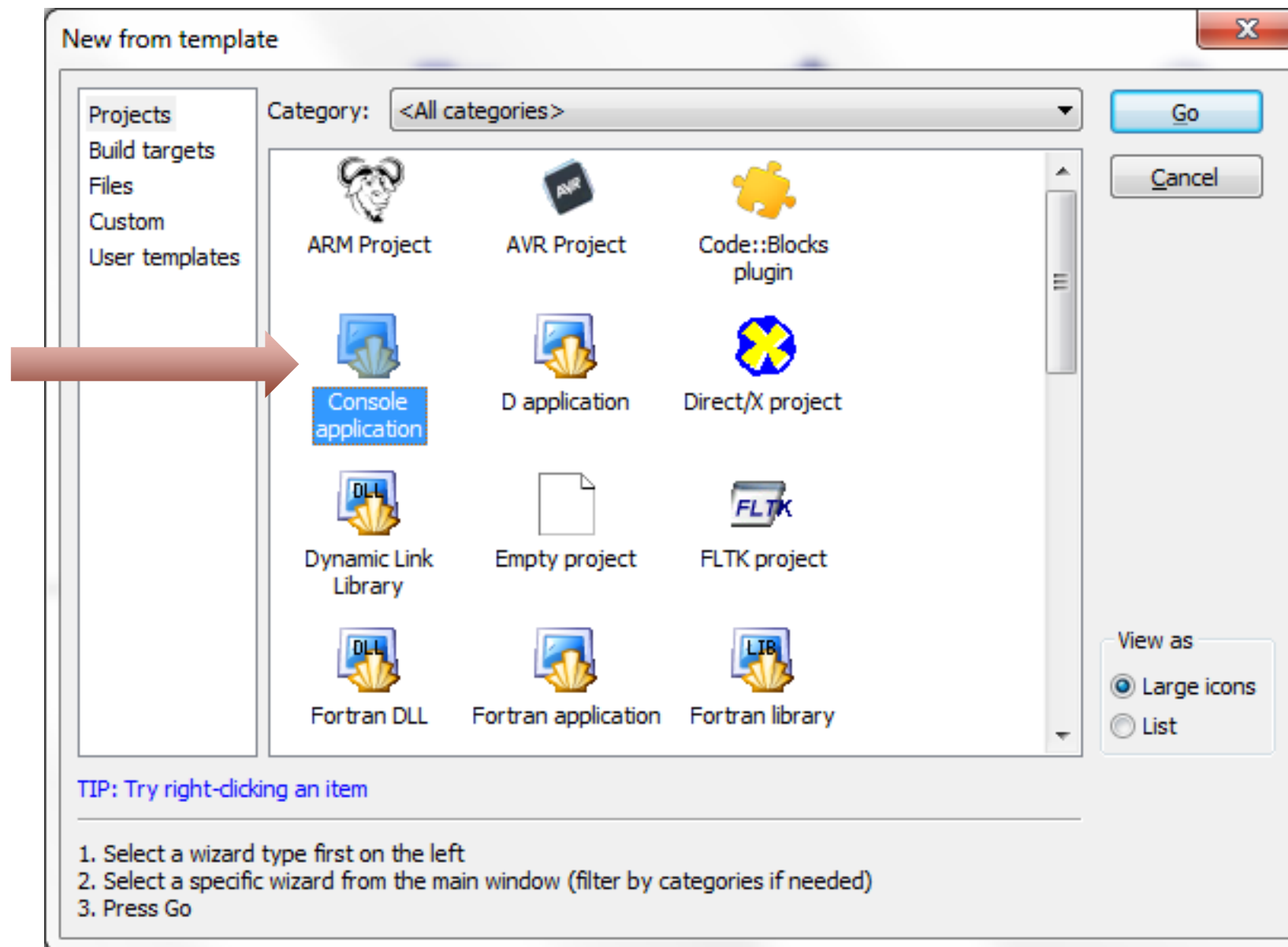
Algoritmos

- **IDE - CodeBlocks**



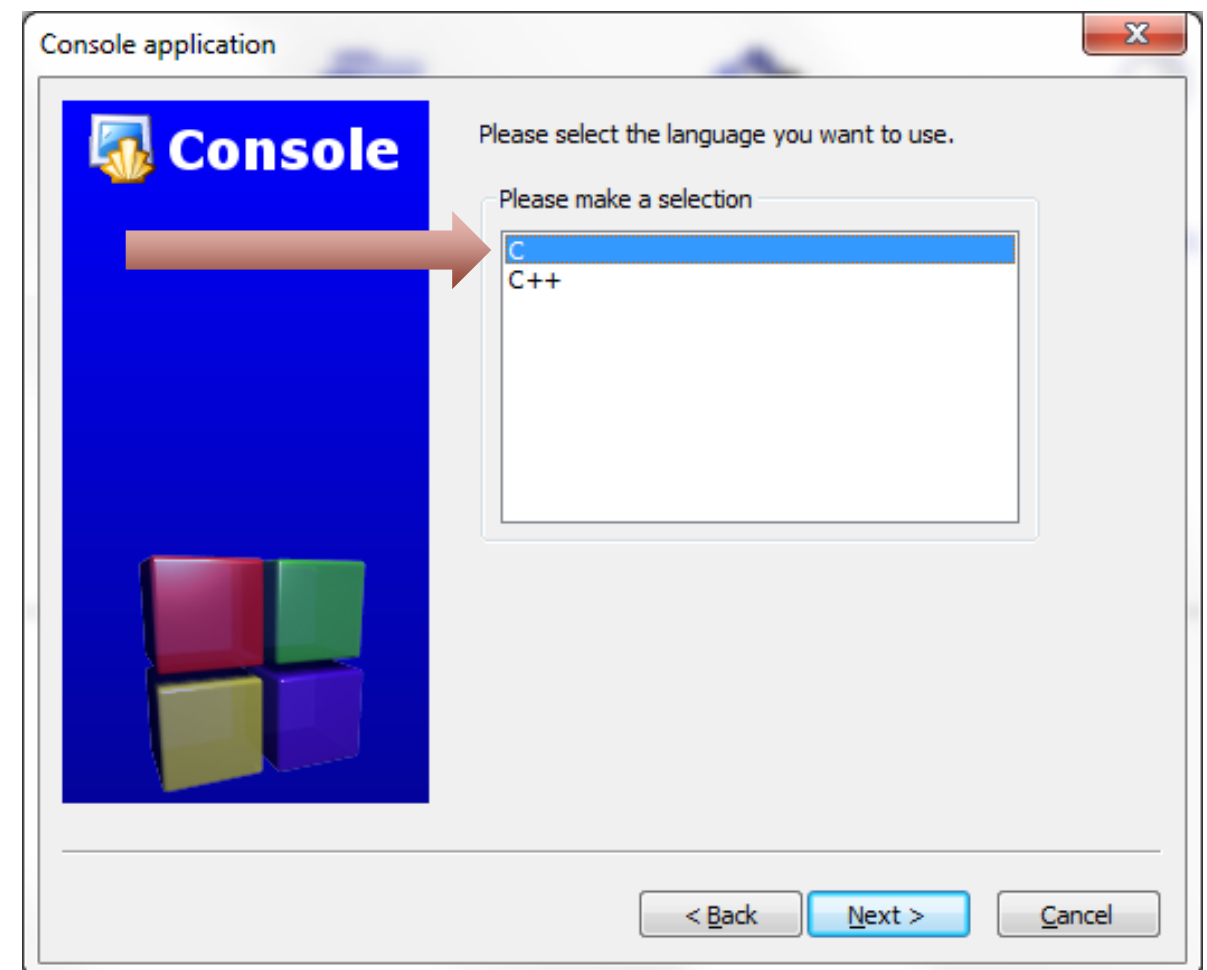
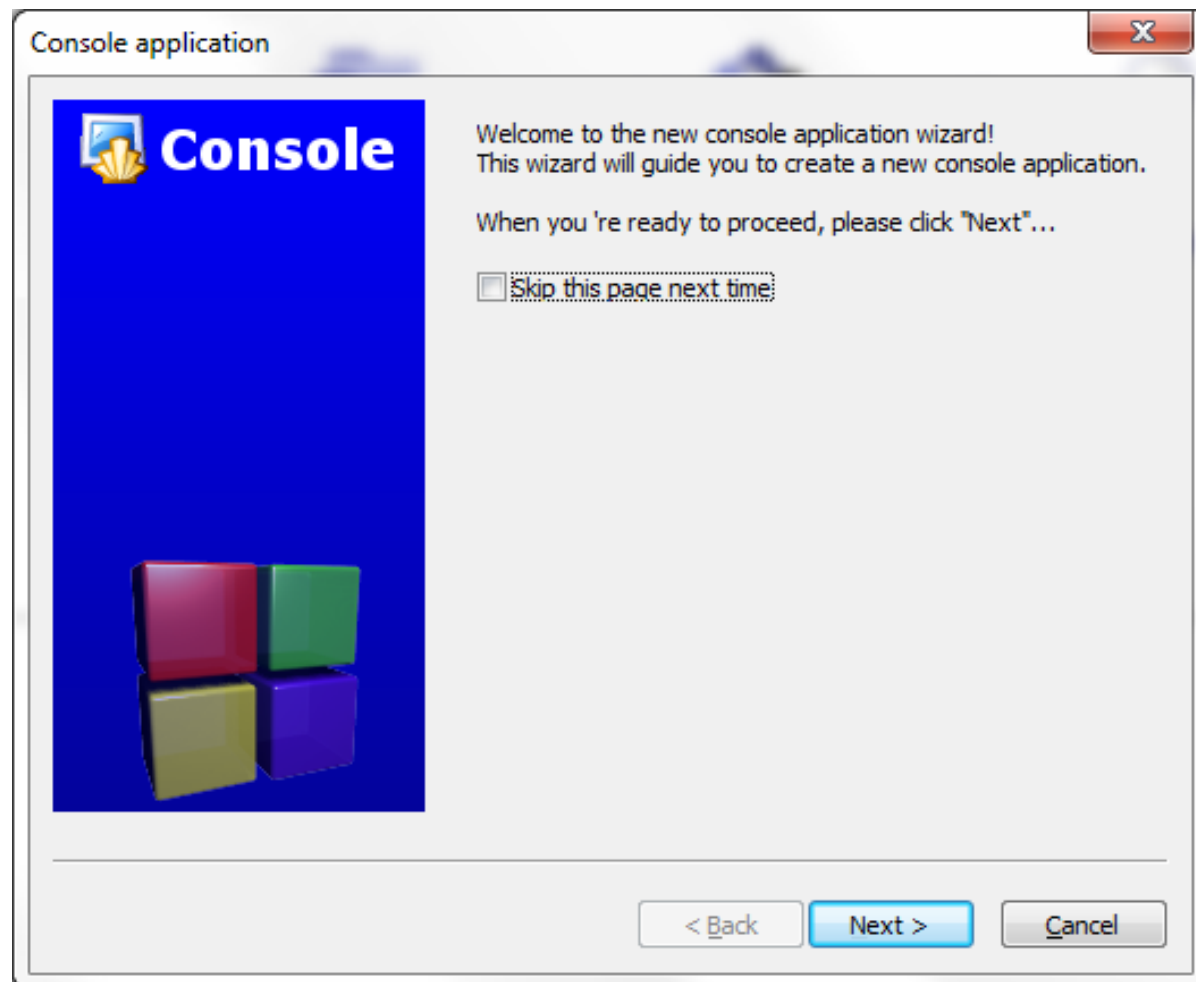
Algoritmos

- **IDE - CodeBlocks**



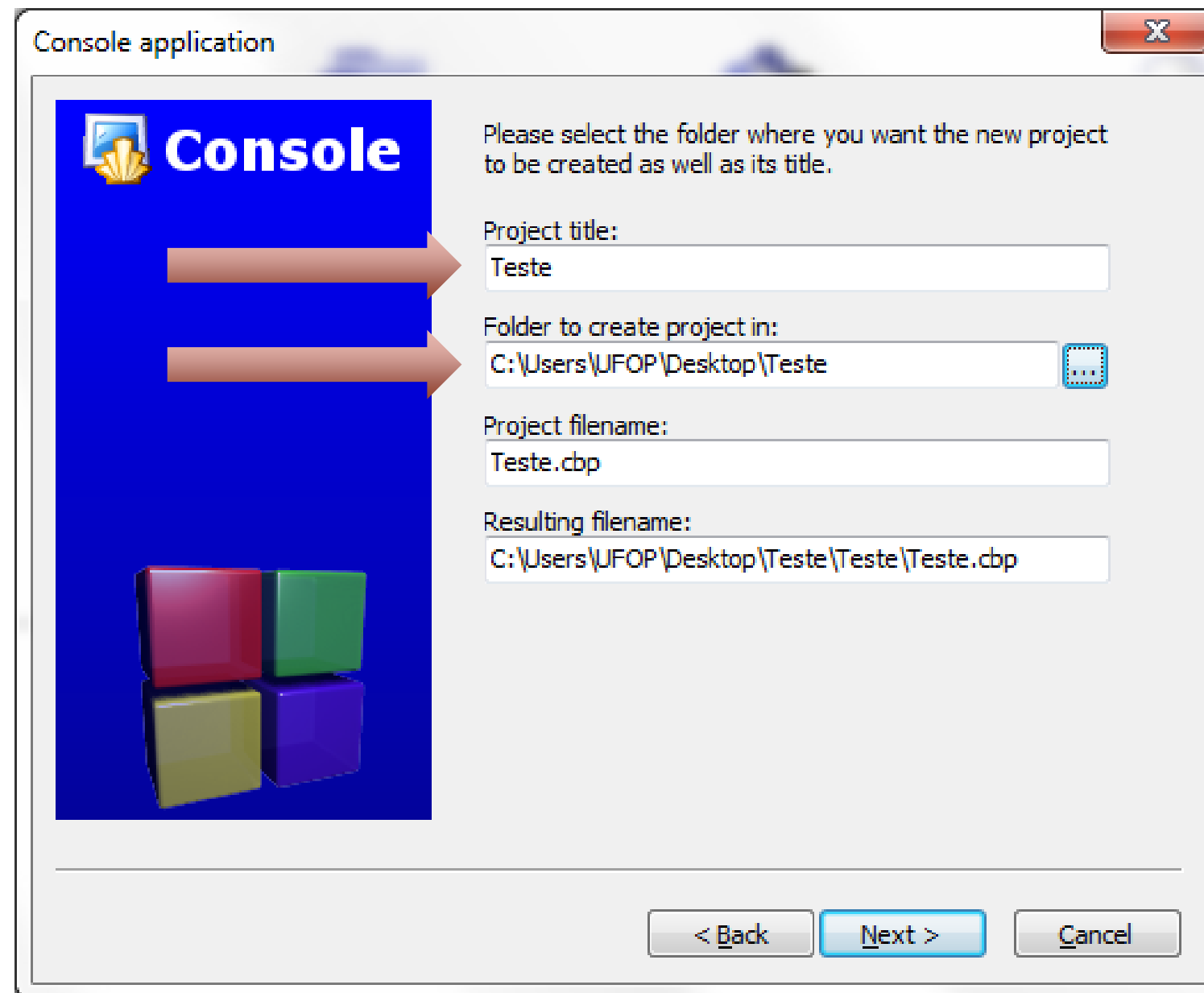
Algoritmos

- **IDE - CodeBlocks**



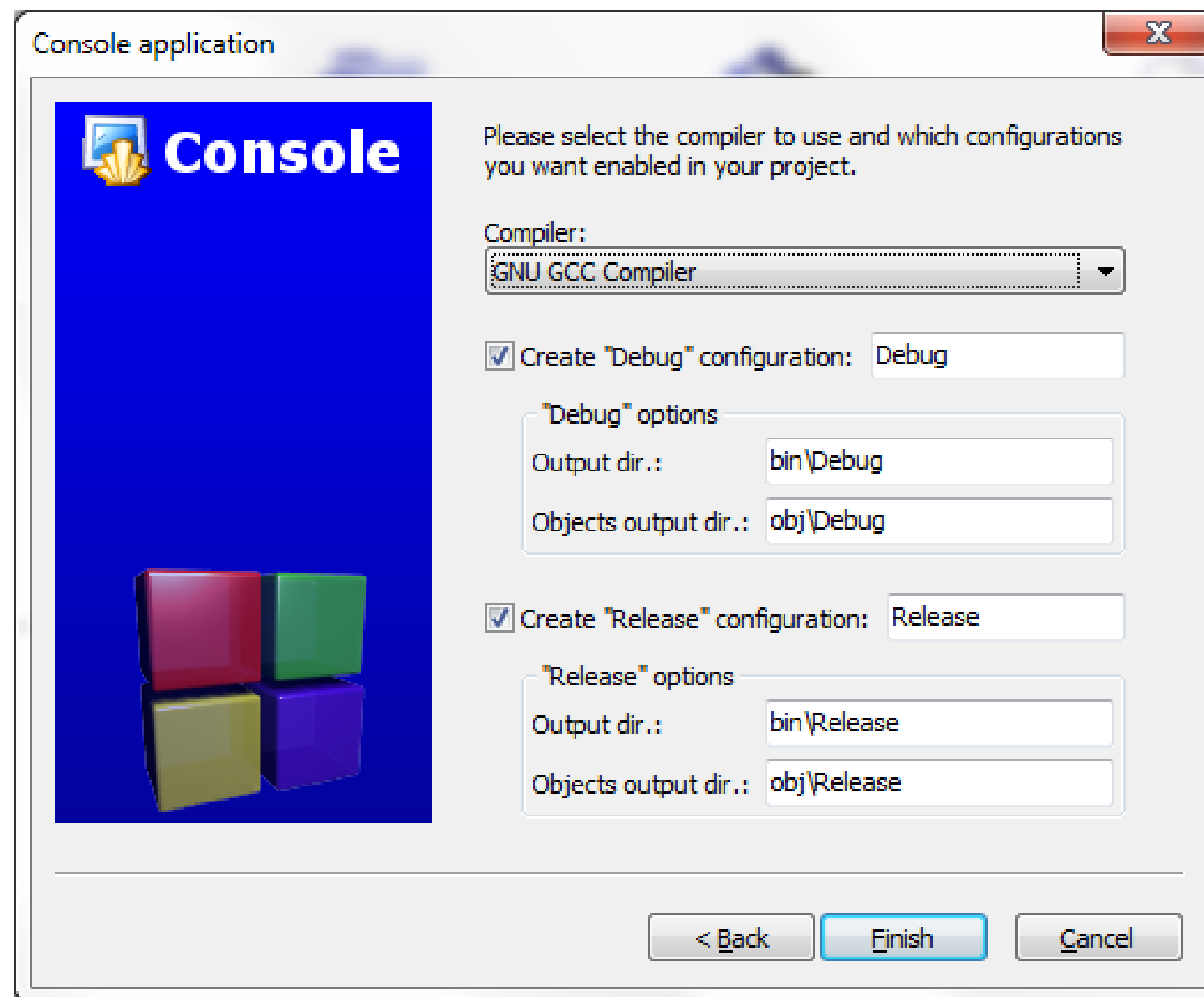
Algoritmos

- **IDE - CodeBlocks**



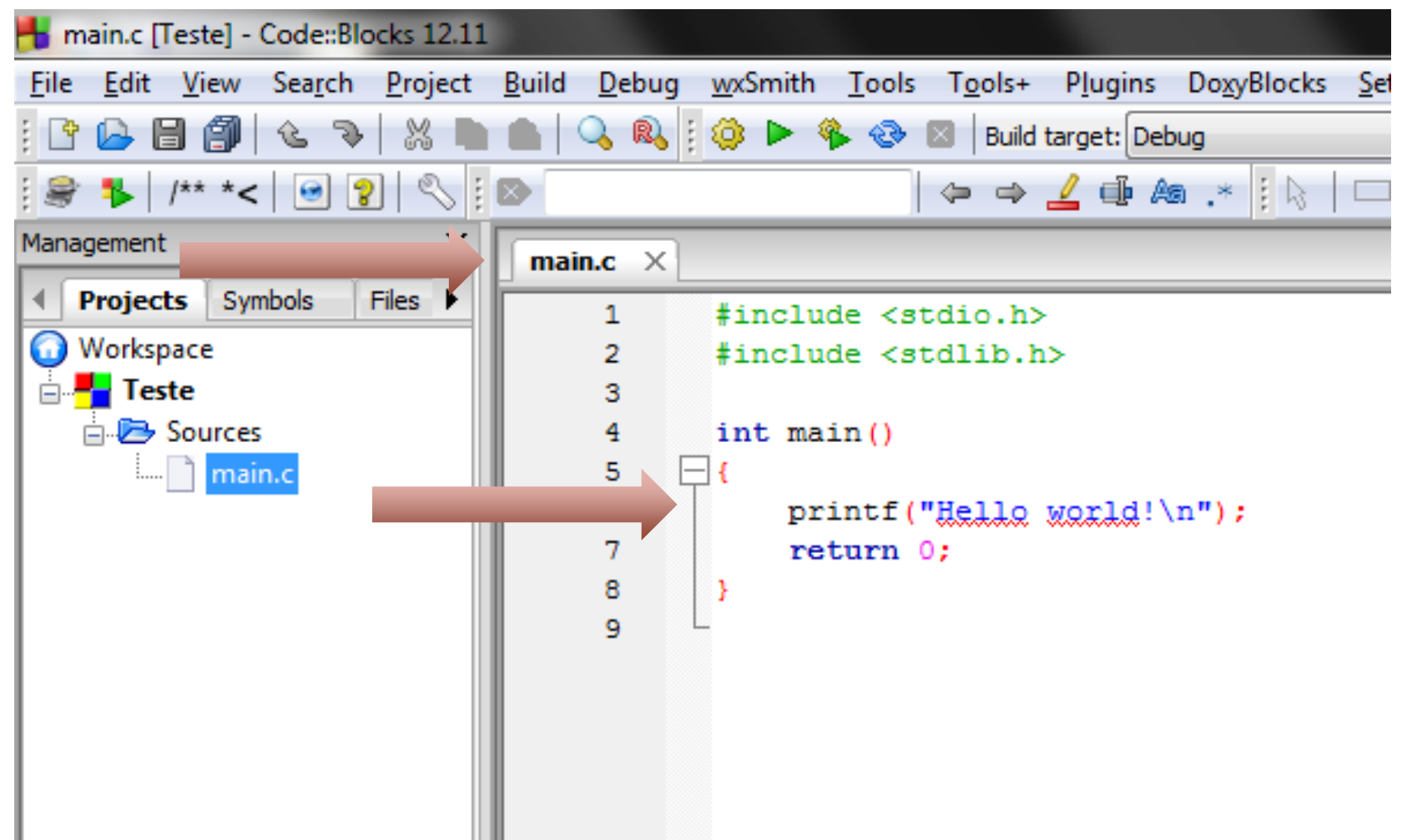
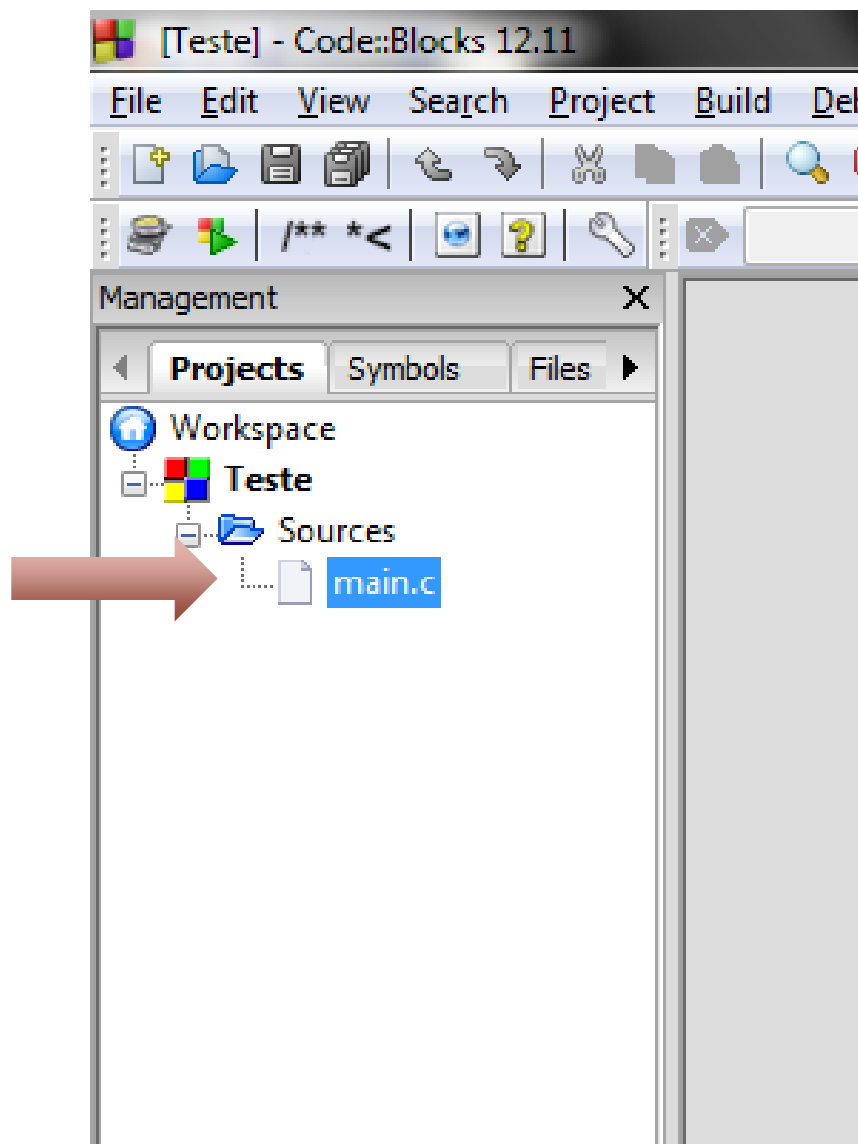
Algoritmos

- **IDE - CodeBlocks**



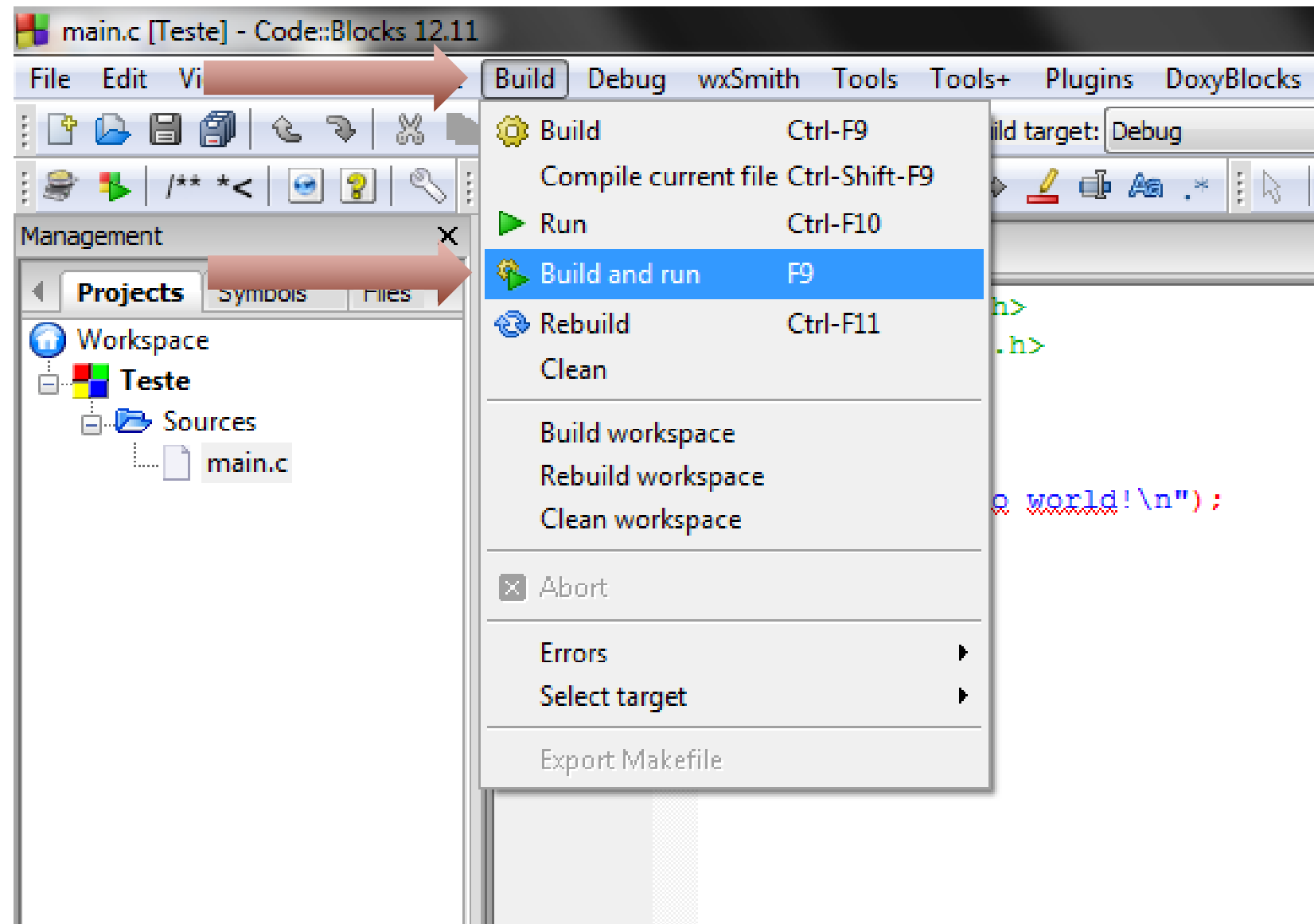
Algoritmos

- **IDE - CodeBlocks**



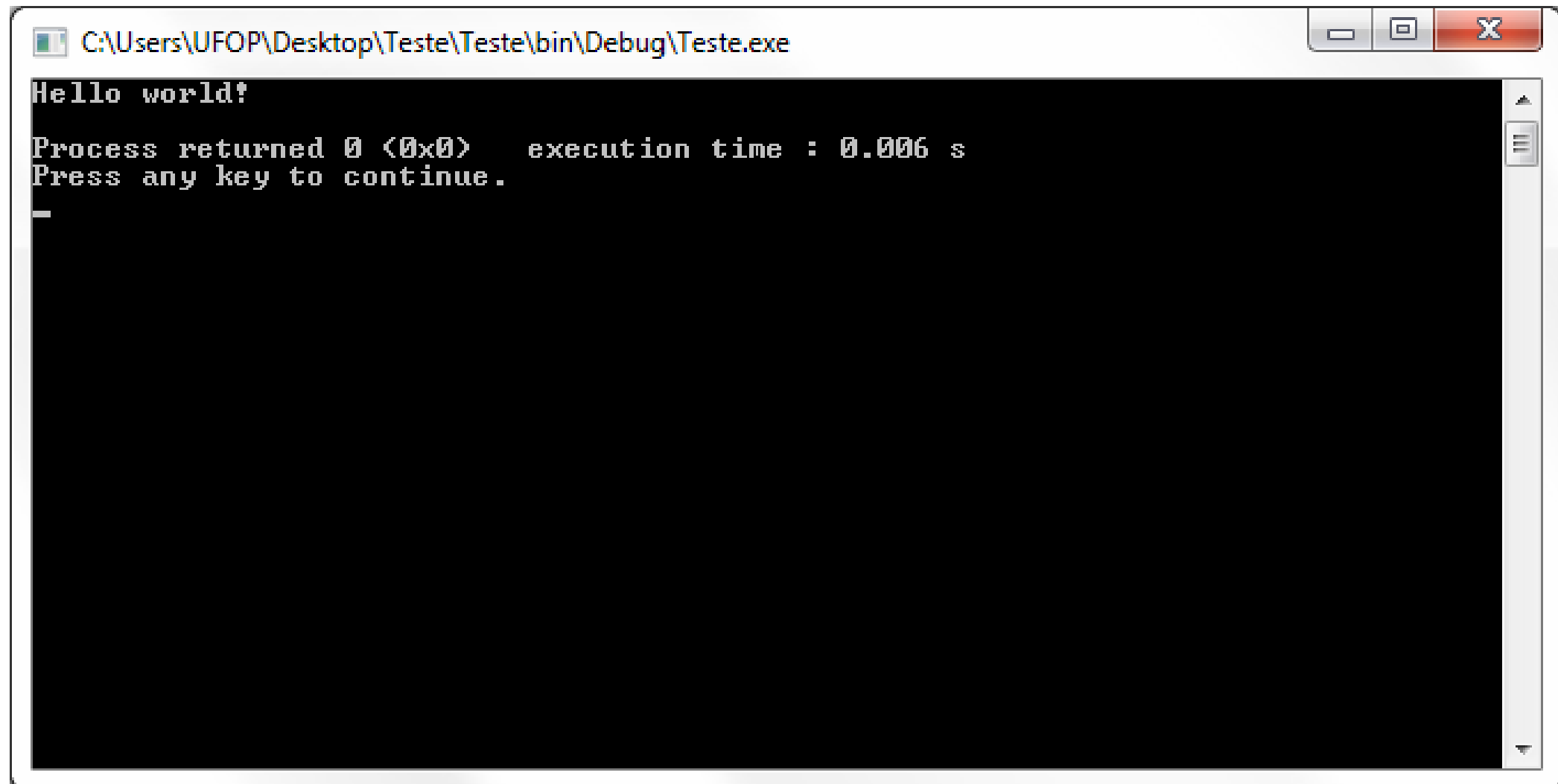
Algoritmos

- **IDE - CodeBlocks**



Algoritmos

- **IDE - CodeBlocks**



A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Users\UFOP\Desktop\Teste\Teste\bin\Debug\Teste.exe" and includes standard minimize, maximize, and close buttons. The main area of the window is black with white text. The text displayed is: "Hello world!" on the first line, "Process returned 0 (0x0) execution time : 0.006 s" on the second line, and "Press any key to continue." on the third line. A single horizontal line is visible on the fourth line, indicating the cursor's position.

```
C:\Users\UFOP\Desktop\Teste\Teste\bin\Debug\Teste.exe
Hello world!
Process returned 0 (0x0) execution time : 0.006 s
Press any key to continue.
_
```


Atividade de Estudo

- Imagine que você foi contratado para desenvolver um robô doméstico que executa tarefas complexas, como lavar a louça, ir ao supermercado, limpar a casa, etc.
- Contudo, este robô apenas atende a comandos simples, como: **ande, vire a direita, levante o braço, localize a porta, etc.**

Atividade de Estudo

- Agora faça o seguinte:
 1. Defina uma lista de comandos simples que seu robô é capaz de realizar.
 2. Escolha 3 tarefas complexas que este robô irá executar e defina a sequencia de comandos simples necessários para a realização desta tarefa.

Referências Bibliográficas

- Material de aula do Prof. Ricardo Anido, da UNICAMP:
<http://www.ic.unicamp.br/~ranido/mc102/>
- Material de aula da Profa. Virgínia F. Mota:
<https://sites.google.com/site/viriniaferm/home/disciplinas>
- DEITEL, P; DEITEL, H. *C How to Program*. 6a Ed. Pearson, 2010.