

# ALLEGRO GAME LIBRARY

Projeto Compartilhar



# Rafael Toledo

- ▶ Bacharel em Sistemas de Informação pela FAI – 2011
- ▶ Pós-Graduado em Desenvolvimento Ágil para Web – FAI 2013
- ▶ Desenvolvedor Android na **MakeMe**
- ▶ Desenvolvedor Android Freelancer na **NtxDev**
- ▶ Instrutor na **Season Treinamentos**
- ▶ **[www.rafaeltoledo.net](http://www.rafaeltoledo.net)**

## QUEM SOU EU?

- ▶ O básico de **linguagem C**
- ▶ Conceitos de Programação de Jogos (Gamedev) usando Allegro 5
  - Imagens
  - Mouse
  - Teclado
  - Animações
  - Música
  - Efeitos Sonoros
  - E o que mais der tempo!

# O QUE TEREMOS NO CURSO?

- ▶ [www.rafaeltoledo.net/tutoriais-alegro-5](http://www.rafaeltoledo.net/tutoriais-alegro-5)
- ▶ [www.rafaeltoledo.net/tutoriais-de-programacao-para-iniciantes](http://www.rafaeltoledo.net/tutoriais-de-programacao-para-iniciantes)
- ▶ [www.programadoresdejogos.com/forum](http://www.programadoresdejogos.com/forum)
- ▶ [www.unidev.com.br/index.php?/index](http://www.unidev.com.br/index.php?/index)
- ▶ [www.cplusplus.com](http://www.cplusplus.com)
- ▶ [www.gamedev.net/index](http://www.gamedev.net/index)
- ▶ [www.google.com](http://www.google.com)

## LINKS ÚTEIS



# PARTE 1 – INTRODUÇÃO A LINGUAGEM C

- ▶ A linguagem C foi projetada para criar programas pequenos e velozes!
- ▶ Nível mais baixo que a maioria das linguagens
- ▶ **Cria um código mais próximo do que o computador entende**

PARA APLICATIVOS PEQUENOS E  
VELOZES!

código

```
#include <stdio.h>

int main()
{
    puts("C Rules!");
    return 0;
}
```



compilador

```
> gcc rules.c -o rules
>
```



executável

```
10010101010
01010100001
01010101001
01111101000
```

rules  
ou  
rules.exe

# COMO C FUNCIONA?

```
int contador_cartas = 11;  
if (contador_cartas > 10)  
    puts("O baralho tá bom!");
```

O QUE O CÓDIGO FAZ?



Cria uma variável  
inteira e atribui 11 a  
ela

```
int contador_cartas = 11;  
if (contador_cartas > 10)  
    puts("O baralho tá bom!");
```

O QUE O CÓDIGO FAZ?

```
int contador_cartas = 11;  
if (contador_cartas > 10)  
    puts("O baralho tá bom!");
```

O contador é maior  
que 10?

O QUE O CÓDIGO FAZ?

```
int contador_cartas = 11;  
if (contador_cartas > 10)  
    puts("O baralho tá bom!");
```

Se for, exiba a  
mensagem no  
console

# O QUE O CÓDIGO FAZ?

Um número inteiro

```
int contador_cartas = 11;  
if (contador_cartas > 10)  
    puts("O baralho tá bom!");
```

Isso mostra uma  
string no console ou  
terminal

# O QUE O CÓDIGO FAZ?

```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```

O QUE O CÓDIGO FAZ?

Cria uma variável  
inteira e atribui 10

```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```

O QUE O CÓDIGO FAZ?

Enquanto o valor for  
positivo...

```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```

O QUE O CÓDIGO FAZ?

```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```

...exiba a  
mensagem...

# O QUE O CÓDIGO FAZ?



```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```

...e decremente o  
contador

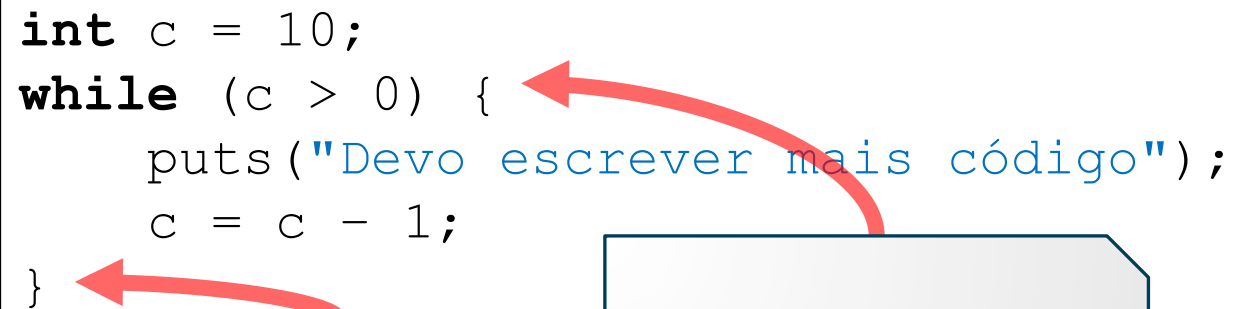
# O QUE O CÓDIGO FAZ?

```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```

Esse é o fim do  
código que deve  
ser repetido

# O QUE O CÓDIGO FAZ?

```
int c = 10;  
while (c > 0) {  
    puts("Devo escrever mais código");  
    c = c - 1;  
}
```



**As chaves definem  
um bloco de código**

# O QUE O CÓDIGO FAZ?

```
/* Assuma um nome com menos de 20 caracteres */  
char ex[20];  
puts("Entre com seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

# O QUE O CÓDIGO FAZ?

Isso é um  
comentário

```
/* Assuma um nome com menos de 20 caracteres */  
char ex[20];  
puts("Entre com seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

# O QUE O CÓDIGO FAZ?

Crie um vetor de 20  
caracteres

```
/* Assuma um nome com menos de 20 caracteres */  
char ex[20];  
puts("Entre com seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

# O QUE O CÓDIGO FAZ?

Mostre uma  
mensagem na tela

```
/* Assume um nome com menos de 20 caracteres */  
char ex[20];  
puts("Entre com seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

# O QUE O CÓDIGO FAZ?

Armazene o que o  
usuário digitar no  
vetor

```
/* Assuma um nome com no máximo de 20 caracteres */  
char ex[20];  
puts("Entre com seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

# O QUE O CÓDIGO FAZ?



Mostre uma  
mensagem  
incluindo o texto  
digitado

```
/* Assuma um nome com 19 caracteres */  
char ex[20];  
puts("Entre com seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

# O QUE O CÓDIGO FAZ?

```
/* Assuma um nome com menos de 20 caracteres */  
char ex[20];  
puts("Entre seu nome: ");  
scanf("%19s", ex);  
printf("Olá, %s!\n\nTudo bem?\n", ex);
```

Isso significa: "Guarde  
tudo que o usuário  
digitar em ex"

Isso vai inserir esta  
string de caracteres  
no lugar do %s

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
  case 'P':  
    puts("Paus");  
    break;  
  case 'O':  
    puts("Ouros");  
    break;  
  case 'C':  
    puts("Copas");  
    break;  
  default:  
    puts("Espadas");  
}
```

Cria uma variável  
do tipo caractere;  
armazena a letra C

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
  case 'P':  
    puts("Paus");  
    break;  
  case 'O':  
    puts("Ouros");  
    break;  
  case 'C':  
    puts("Copas");  
    break;  
  default:  
    puts("Espadas");  
}
```

Olhe para o valor  
da variável

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

É igual a P?

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

Se for, escreva a  
palavra "Paus"

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

E pule as outras  
verificações

# O QUE O CÓDIGO FAZ?



```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

É igual a O?

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

Se for, escreva a  
palavra "Ouros"

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

E pule as outras  
verificações

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

É igual a C?

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

Se for, escreva a  
palavra "Ouros"

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espada");  
}
```

E pule as outras  
verificações

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

Em último caso...

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

Mostre a palavra  
"Espadas"

# O QUE O CÓDIGO FAZ?



Fim das  
verificações

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

# O QUE O CÓDIGO FAZ?

```
char naipe = 'C';  
switch (naipe) {  
case 'P':  
    puts("Paus");  
    break;  
case 'O':  
    puts("Ouros");  
    break;  
case 'C':  
    puts("Copas");  
    break;  
default:  
    puts("Espadas");  
}
```

Um switch verifica o valor de uma única variável contra uma série de valores

# O QUE O CÓDIGO FAZ?

E COMO É UM PROGRAMA EM C  
COMPLETO?



```
/*
 * Programa para calcular o número de cartas em um balde.
 * Este código é disponibilizado pela Licença Pública Inútil
 * (c)2014, Equipe Contadora de Cartas
 */
#include <stdio.h>

int main()
{
    int baralhos;
    puts("Entre com o número de baralhos:");
    scanf("%i", &baralhos);
    if (baralhos < 1) {
        puts("Não é um número válido de baralhos");
        return 1;
    }
    printf("Existem %i cartas\n", (baralhos * 52));
    return 0;
}
```


**Programas geralmente começam com um comentário.  
Não é obrigatório, mas é uma boa prática!**

```
/*  
 * Programa para calcular o número de cartas em um balde.  
 * Este código é disponibilizado pela Licença Pública Inútil  
 * (c)2014, Equipe Contadora de Cartas  
 */
```

**Um comentário  
começa com /\***



```
/*  
 * Programa para calcular o número de cartas em um balde.  
 * Este código é disponibilizado pela Licença Pública Inútil  
 * (c)2014, Equipe Contadora de Cartas  
 */
```



```
/*  
 * Programa para calcular o número de cartas em um balde.  
 * Este código é disponibilizado pela Licença Pública Inútil  
 * (c)2014, Equipe Contadora de Cartas  
 */
```



**E termina com \*/**

```
/*  
* Programa para calcular o número de cartas em um balde.  
* Este código é disponibilizado pela Licença Pública Inútil  
* (c)2014, Equipe Contadora de Cartas  
*/
```



**Esses \* são só pra  
deixar mais  
arrumadinho!!!**



Em seguida vem os includes! Como C é uma linguagem muito, mas muito pequena, não dá pra fazer quase nada sem as bibliotecas. Você precisa dizer ao compilador qual código externo usar através dos includes

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

A stdio por exemplo contém código que permite que você  
leia e escreva dados no console

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

Por fim, você vai encontrar as funções no arquivo-fonte. Todo código em C roda em funções. A mais importante delas, que você vai encontrar em todos os programas é a função `main()`. Ela é o ponto de partida no seu programa!

```
int main()
{
    int baralhos;
    puts("Entre com o número de baralhos:");
    scanf("%i", &baralhos);
    if (baralhos < 1) {
        puts("Não é um número válido de baralhos");
        return 1;
    }
    printf("Existem %i cartas\n", (baralhos * 52));
    return 0;
}
```

Igual ==

Diferente !=

Maior que >

Menor que <

Maior ou igual >=

Menor ou igual <=

COMPARADORES

Não !

Ou ||

E &&

OPERADORES LÓGICOS

COMPARATIVO C - PASCAL



Em Pascal

`integer`

`real`

`char`

`boolean`

`string`

Em C

`int`

`float` ou `double`

`char`

`bool` \*

`char[]`

TIPOS

## Em Pascal

```
if (idade <> 18) then  
  begin  
    ...  
  end  
else  
  begin  
    ...  
  end;  
end;
```

## Em C

```
if (idade != 18) {  
  ...  
} else {  
  ...  
}
```

CONDICIONAL 'SE'



## Em Pascal

```
case voto of  
  1: can1 := can1 + 1;  
  2: can2 := can2 + 1;  
  else nulos := nulos + 2;  
end;
```

## Em C

```
switch (voto) {  
  case 1:  
    can1++;  
    break;  
  case 2:  
    can2++;  
    break;  
  default:  
    nulos += 2;  
}
```

# CONDICIONAL 'ESCOLHA'

## Em Pascal

```
while cont < 18 do  
begin  
    cont := cont - 1;  
end;
```

## Em C

```
while (cont < 18) {  
    cont--;  
}
```

REPETIÇÃO 'ENQUANTO'

## Em Pascal

```
for indice := 1 to 10 do  
begin  
    readln(vetor[indice]);  
end;
```

## Em C

```
for (indice = 0; indice < 10; indice++) {  
    scanf("%i", &vetor[indice]);  
}
```

# REPETIÇÃO 'PARA'

## Em Pascal

```
for indice := 1 to 10 do  
begin  
    readln(vetor[indice]);  
end;
```

## Em C

```
for (indice = 0; indice < 10; indice++) {  
    scanf("%i", &vetor[indice]);  
}
```

# REPETIÇÃO 'PARA'



**Vetores em C  
começam em  
0**

Em Pascal

```
repeat
```

```
    x := x / 4;
```

```
until (x < 200);
```

Em C

```
do {
```

```
    x /= 4;
```

```
} while (x >= 200);
```

REPETIÇÃO 'REPITA'

Em Pascal

```
type  
  aluno = record  
    nome: string[30];  
    idade: integer;  
  end;
```

Em C

```
typedef struct {  
    char nome[30];  
    int idade;  
} aluno;
```

REGISTROS – COM CRIAÇÃO DE TIPO

Em Pascal

```
type  
  aluno = record  
    nome: string[30];  
    idade: integer;  
  end;
```

```
typedef struct {  
    char nome[30];  
    int idade;  
} aluno;  
  
struct aluno {  
    char nome[30];  
    int idade;  
};
```

Em C

# REGISTROS

## Em Pascal

```
function somar(n1, n2 : integer) : integer;  
var  
    resultado: integer;  
  
begin  
    resultado := n1 + n2;  
    somar := resultado;  
end;
```

## FUNÇÕES



Em C

```
int somar(int n1, int n2) {  
    int resultado = n1 + n2;  
    return resultado;  
}
```

# FUNÇÕES

Em Pascal

```
procedure somar(n1, n2 : integer);  
var  
    resultado: integer;  
  
begin  
    resultado := n1 + n2;  
    writeln('O resultado é ', resultado);  
end;
```

# PROCEDIMENTOS

Em C

```
void somar(int n1, int n2) {  
    int resultado = n1 + n2;  
    printf("O resultado é %i\n", resultado);  
}
```

# PROCEDIMENTOS

Um procedimento em C nada mais é que  
uma função que diz que não vai retornar  
nada – tipo **void**

Em C

```
void somar(int n1, int n2) {  
    int resultado = n1 + n2;  
    printf("O resultado é %i\n", resultado);  
}
```

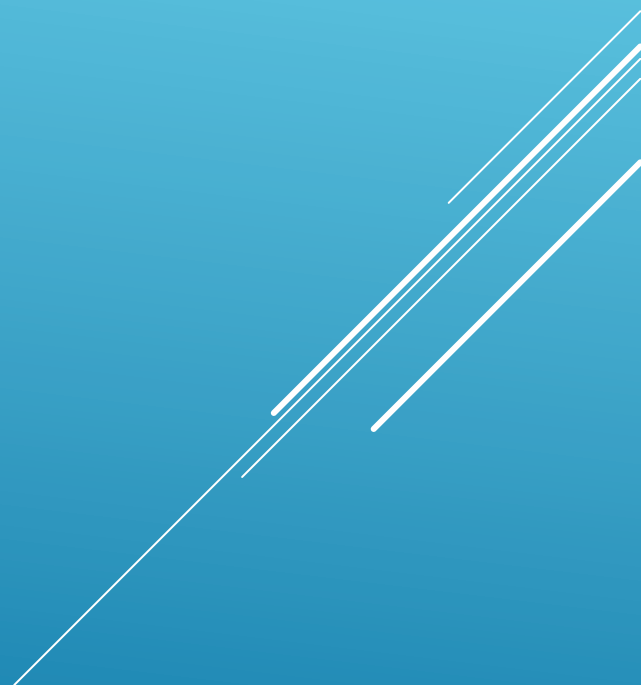
# PROCEDIMENTOS

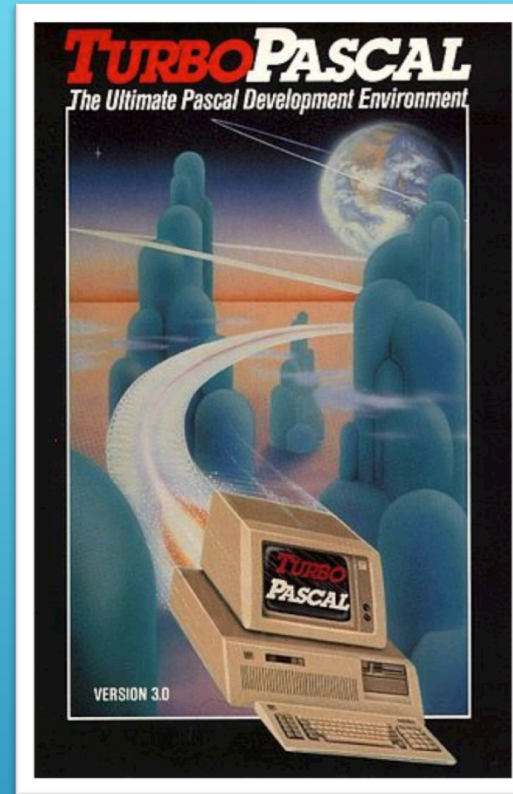
▶ C, ao contrário de Pascal, é **CASE SENSITIVE!**

▶ **variavel** é diferente de **VARIAVEL**

OBSERVAÇÃO IMPORTANTE

E ONDE VAMOS DESENVOLVER?



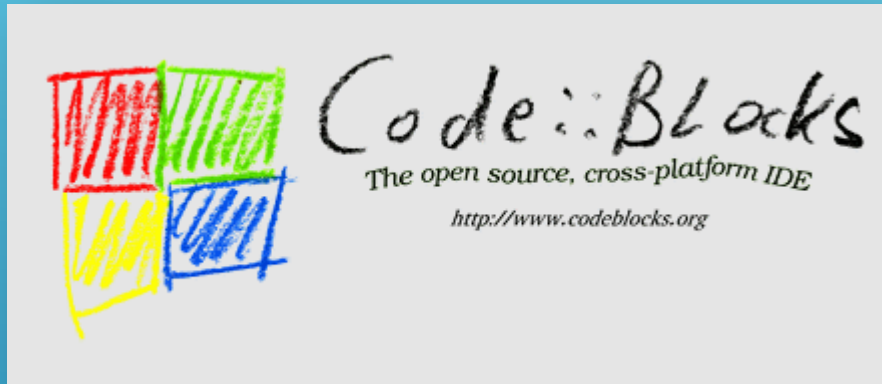


E ONDE VAMOS DESENVOLVER?

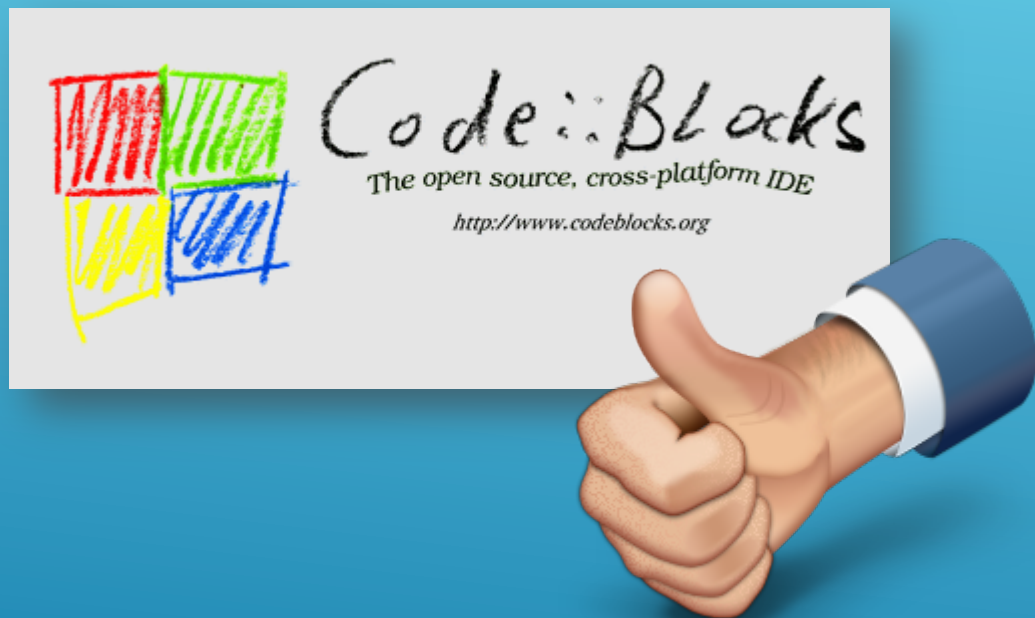


E ONDE VAMOS DESENVOLVER?

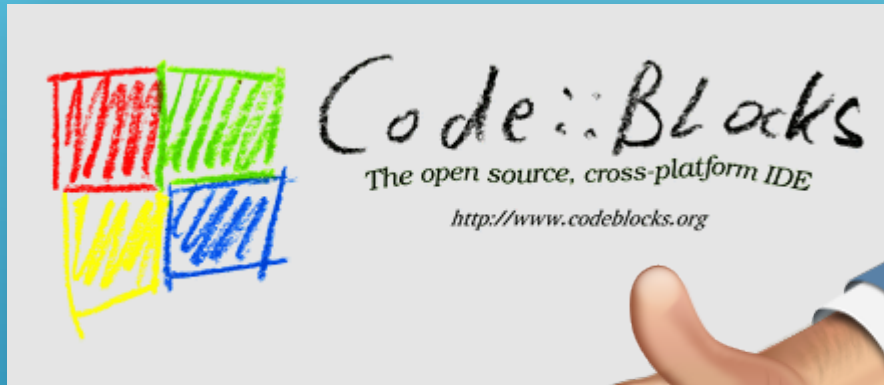




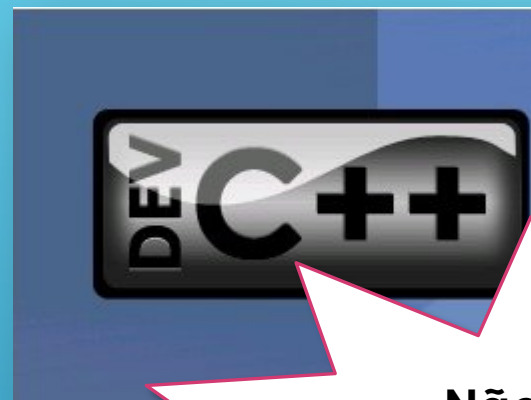
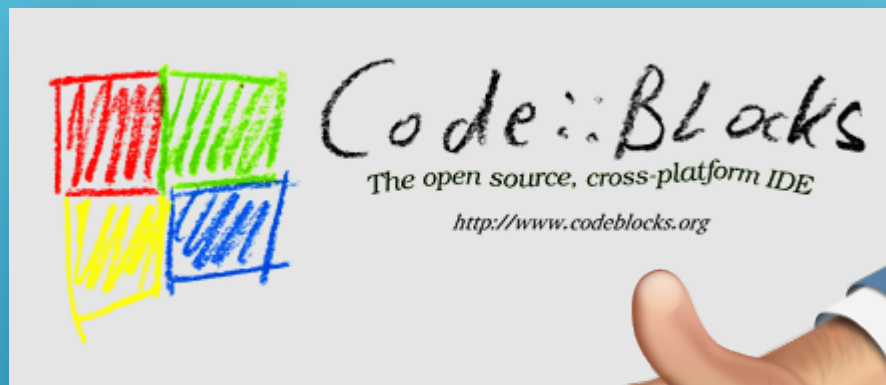
E ONDE VAMOS DESENVOLVER?



E ONDE VAMOS DESENVOLVER?



E ONDE VAMOS DESENVOLVER?



Não tem  
atualizações  
desde 2005!!!

E ONDE VAMOS DESENVOLVER?

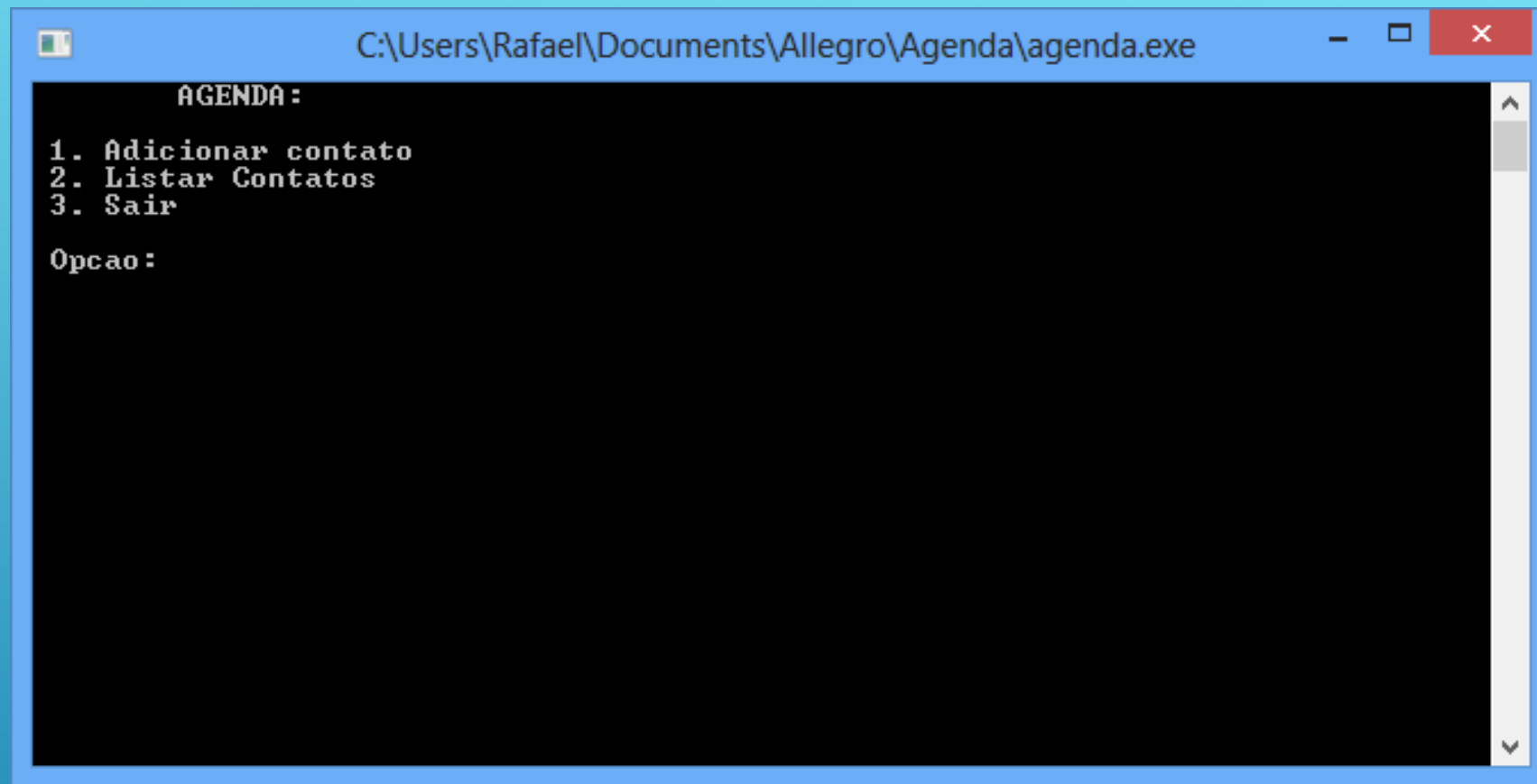
▶Pra baixar o Code::Blocks

▶[link](#)

DOWNLOAD DO CODEBLOCKS



MÃOS À OBRA?



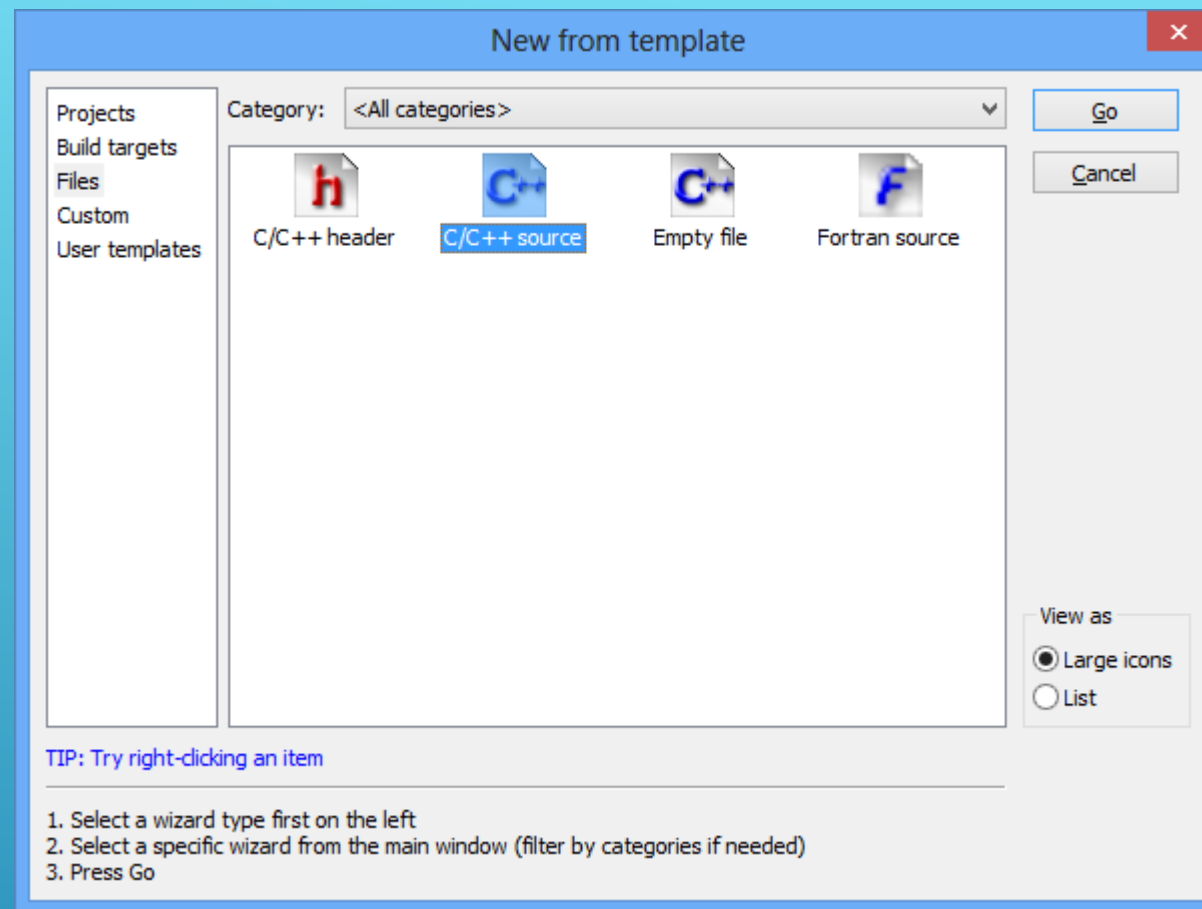
# PROJETO #1 – AGENDA!



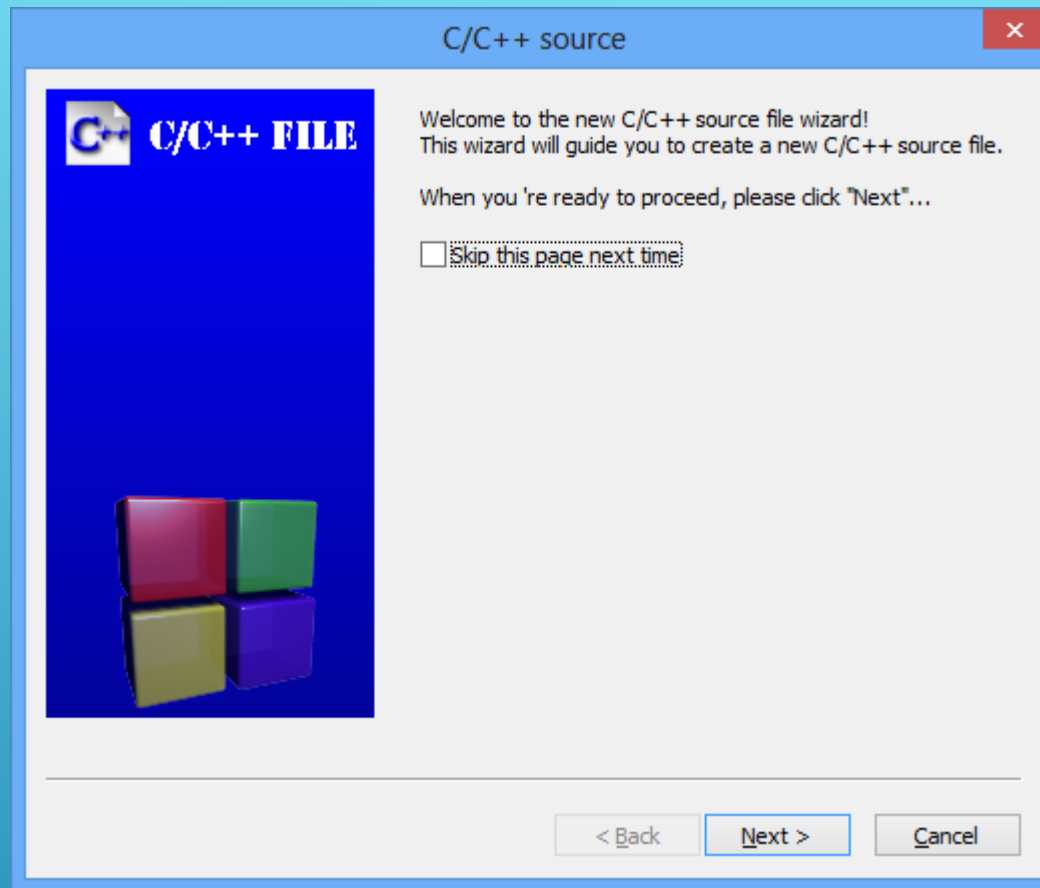
- ▶ Abrir o Code::Blocks
- ▶ **File -> New -> File...**

PROJETO #1 – AGENDA!

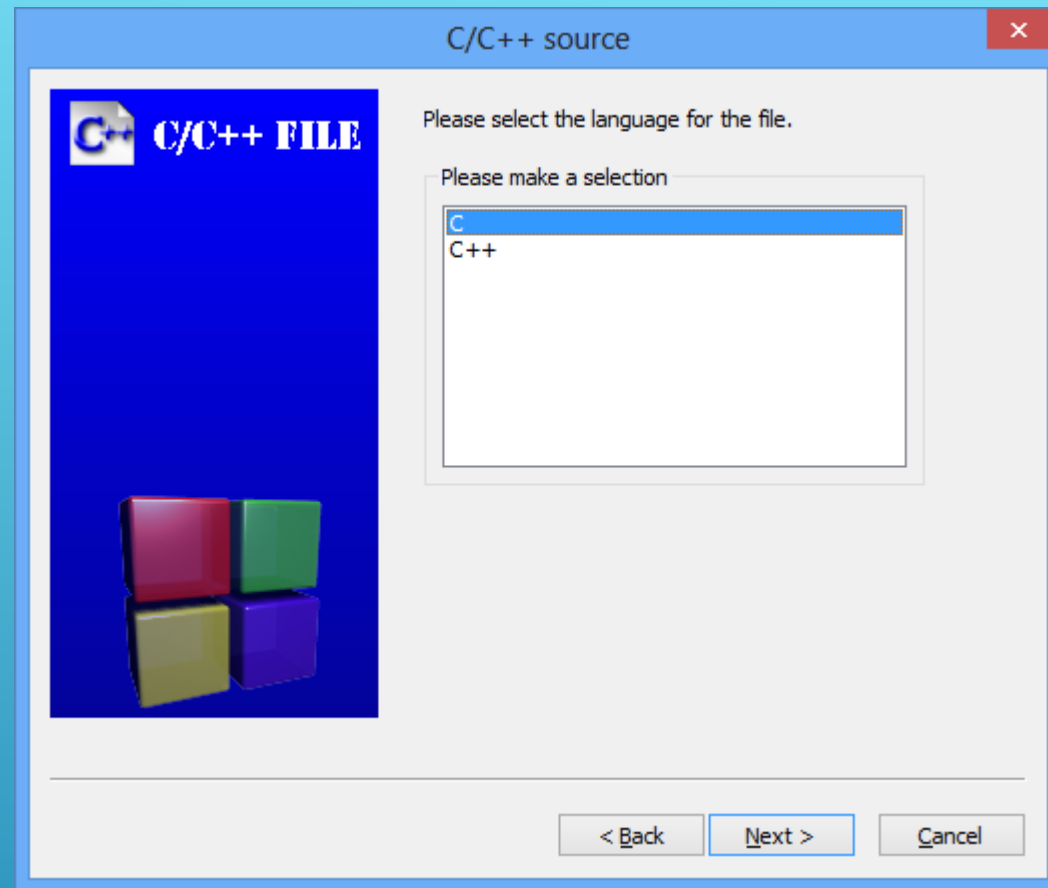




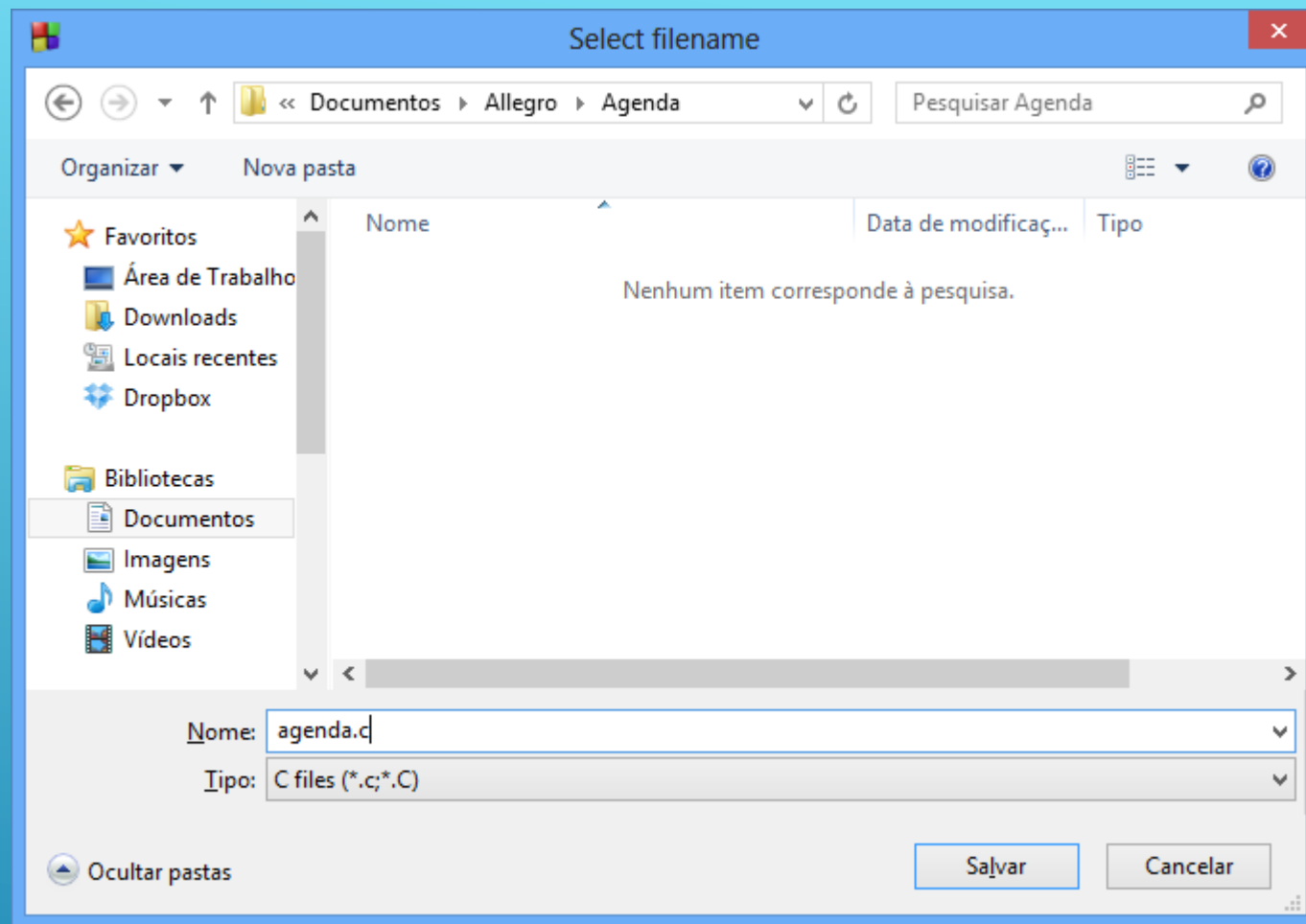
# PROJETO #1 – AGENDA!



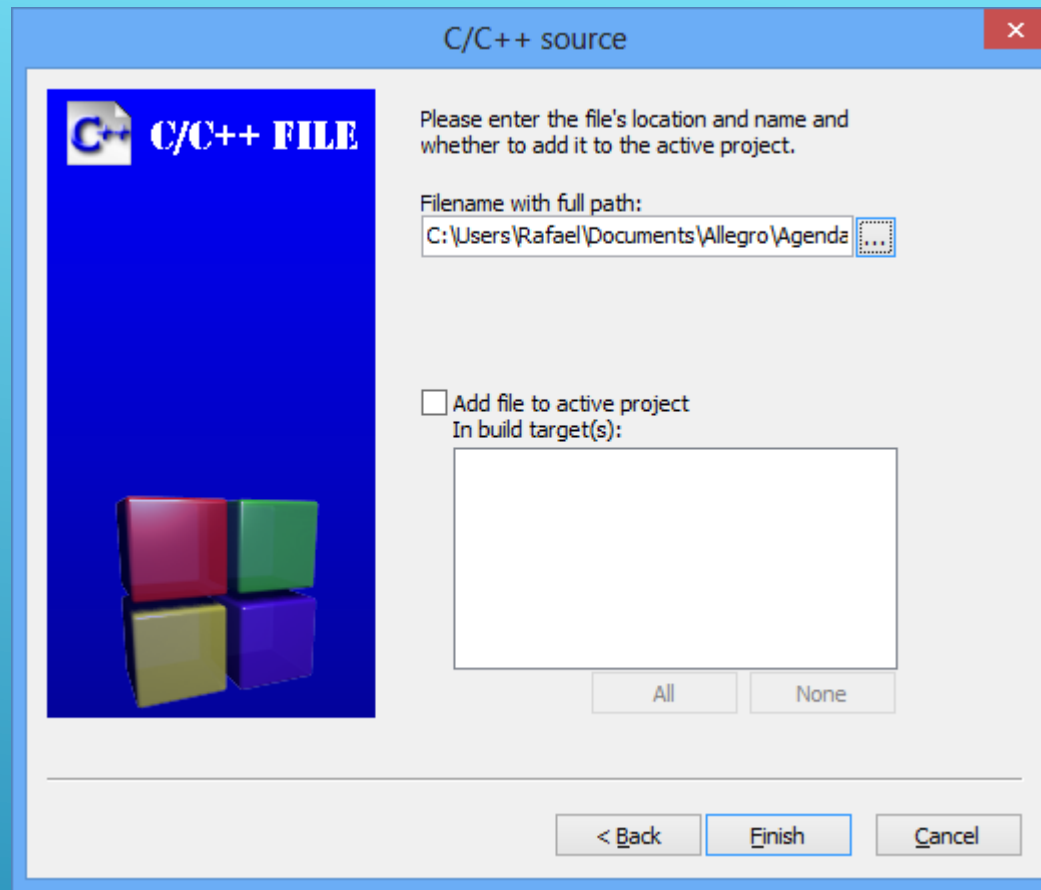
# PROJETO #1 – AGENDA!



PROJETO #1 – AGENDA!



# PROJETO #1 – AGENDA!



# PROJETO #1 – AGENDA!

- ▶ Compilar e rodar: **F9**
- ▶ Somente compilar: **Ctrl + F9**

PROJETO #1 – AGENDA!



## PARTE 2 – CONHECENDO A ALLEGRO

- Desenvolvida originalmente por Shawn Hargreaves para utilização no Atari ST, lá em 1987



SOBRE A BIBLIOTECA ALLEGRO



- ▶ Com a descontinuação do projeto, Hargreaves portou o código para o Borland C e para o DJGPP
- ▶ Porém o Borland C foi descontinuado em sua versão 2.0...

***azarado o cara hein....***

SOBRE A BIBLIOTECA ALLEGRO

- ▶ Os jogos começaram então a ser produzidos para a plataforma **MS-DOS**
- ▶ Em 1998 a biblioteca ganhou versões para **Windows** (WinAllegro) e **Linux** (XwinAllegro)
- ▶ A versão **4** foi a primeira oficialmente multiplataforma

SOBRE A BIBLIOTECA ALLEGRO

- ▶ Plataformas suportadas pela versão 5:  
**Linux, Windows, OSX e iOS**
- ▶ A versão 5 foi um grande passo para a evolução da biblioteca
- ▶ Foi realizada uma grande reestruturação em sua API

**SOBRE A BIBLIOTECA ALLEGRO**

- ▶ Em contrapartida, tornou-se incompatível com versões anteriores
- ▶ **Evita a necessidade de utilizar-se diversos plug-ins, externos a biblioteca. Dividida em add-ons**

SOBRE A BIBLIOTECA ALLEGRO

- ▶ Allegro Main
- ▶ Allegro Image
- ▶ Allegro Primitives
- ▶ Allegro Color
- ▶ Allegro Font
- ▶ Allegro TTF
- ▶ Allegro Audio
- ▶ Allegro Acodec
- ▶ Allegro Memfile
- ▶ Allegro PhysFS
- ▶ Allegro Native Dialog

ADD-ONS

- ▶ Não é necessário *linkar* o projeto com todos eles. Assim fica **mais rápido** pra compilar e **mais leve** pra distribuir!

QUAL A VANTAGEM DE ADD-ONS?

- ▶ Para o curso, vamos utilizar a versão mais recente disponível, a **5.0.10** em parceria com o **Code::Blocks 13.12**

VERSÕES UTILIZADAS

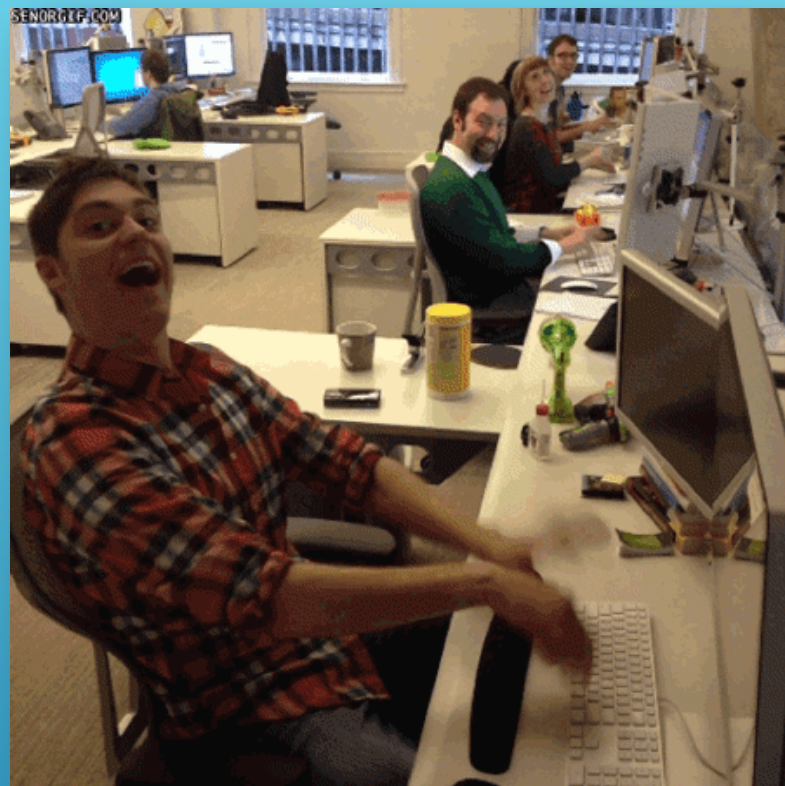
- ▶ Baixar o arquivo daqui (zip ou 7-zip)
- ▶ Descompactar, copiar as pastas **bin**, **lib** e **include** e colar em  
**C:\Program Files\CodeBlocks\MinGW**  
OU  
**C:\Program Files (x86)\CodeBlocks\MinGW**

COMO INSTALAR A ALLEGRO?



E PRONTO!

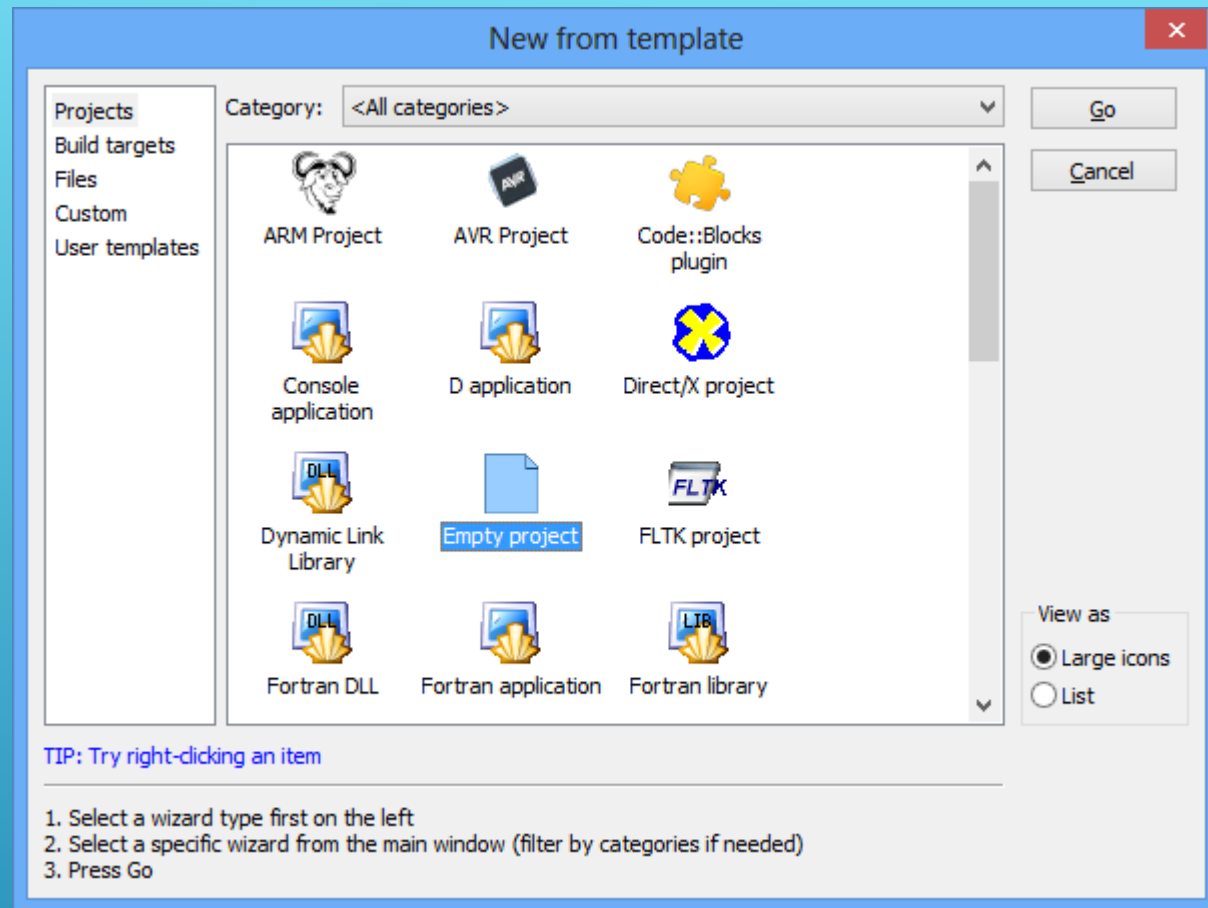




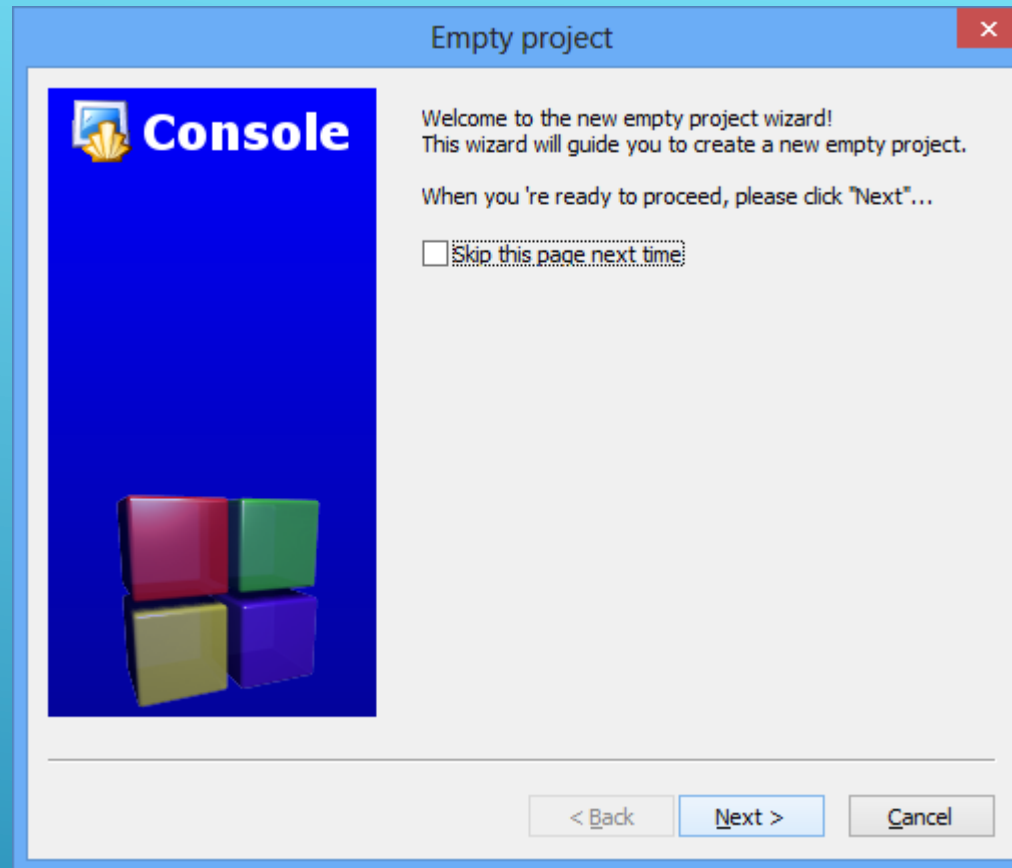
TODOS PRONTOS PRA MAIS UMA?

- ▶ Abrir o Code::Blocks
- ▶ **File -> New -> Project...**

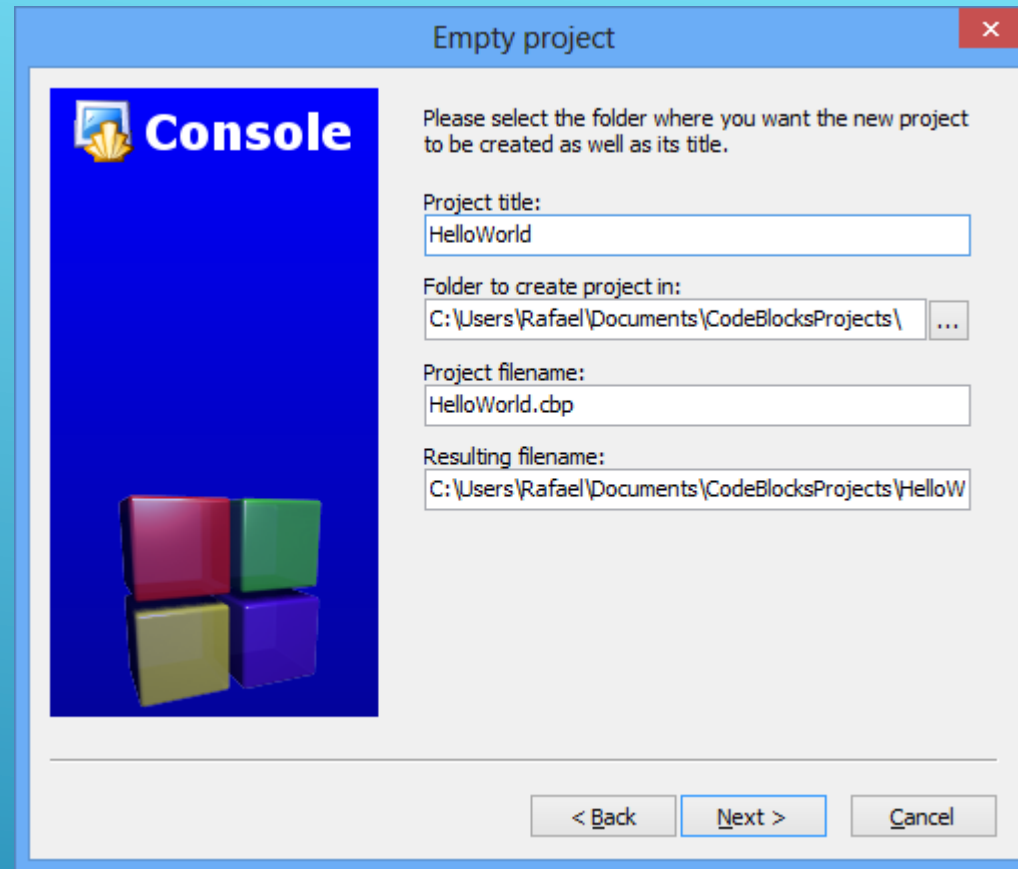
PROJETO #2 – HELLO WORLD ALLEGRO!



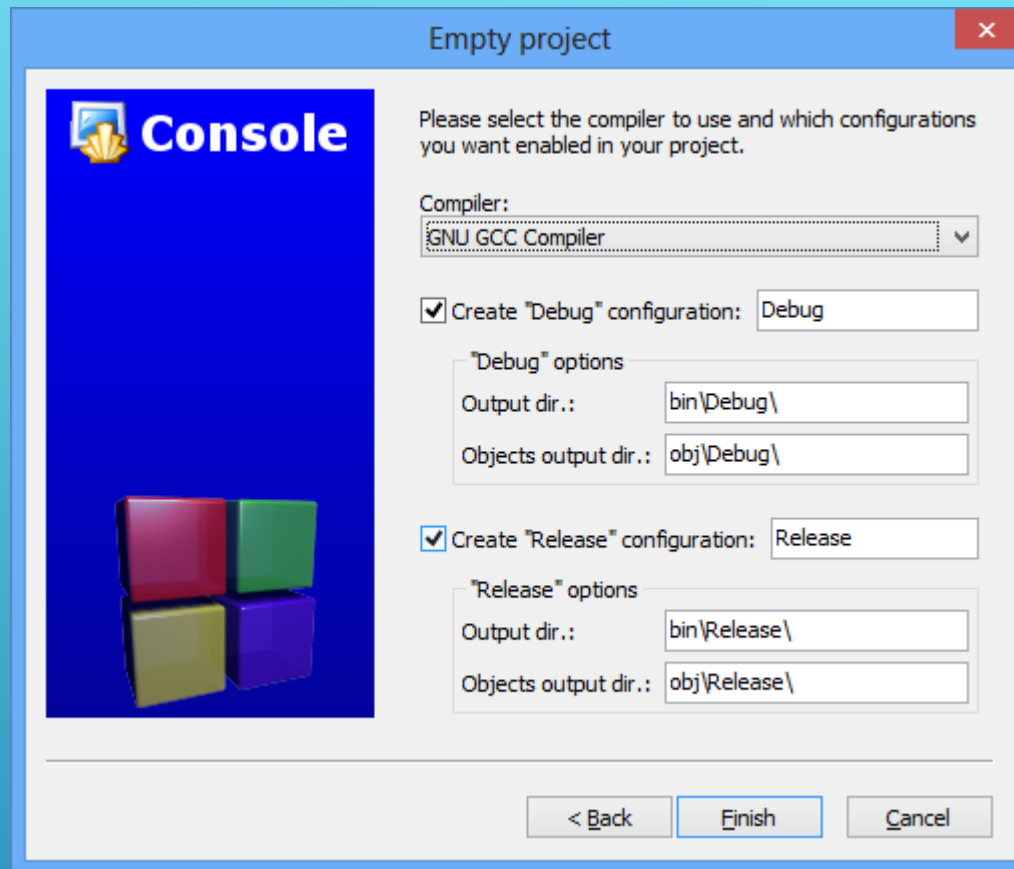
## PROJETO #2 – HELLO WORLD ALLEGRO!



# PROJETO #2 – HELLO WORLD ALLEGRO!



PROJETO #2 – HELLO WORLD ALLEGRO!



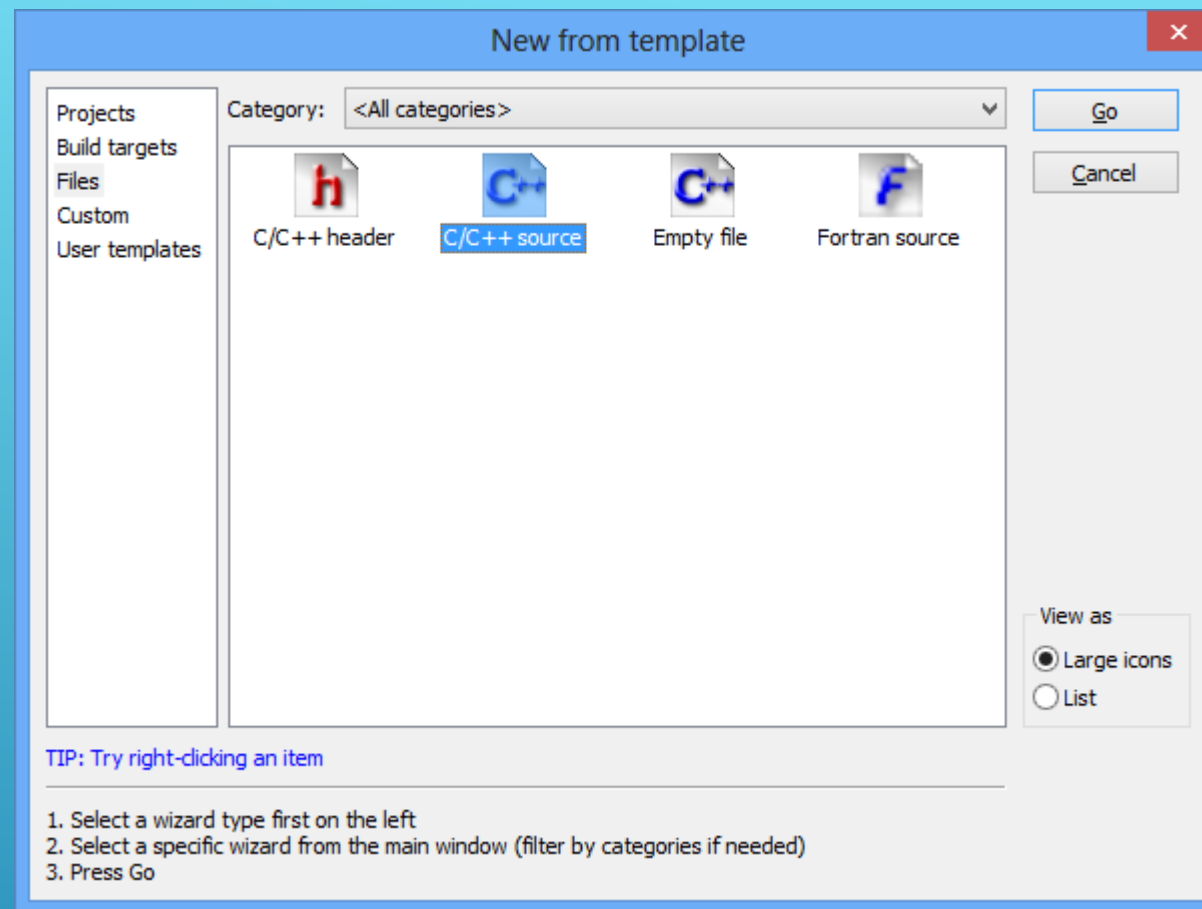
# PROJETO #2 – HELLO WORLD ALLEGRO!

► **File -> New -> File...**

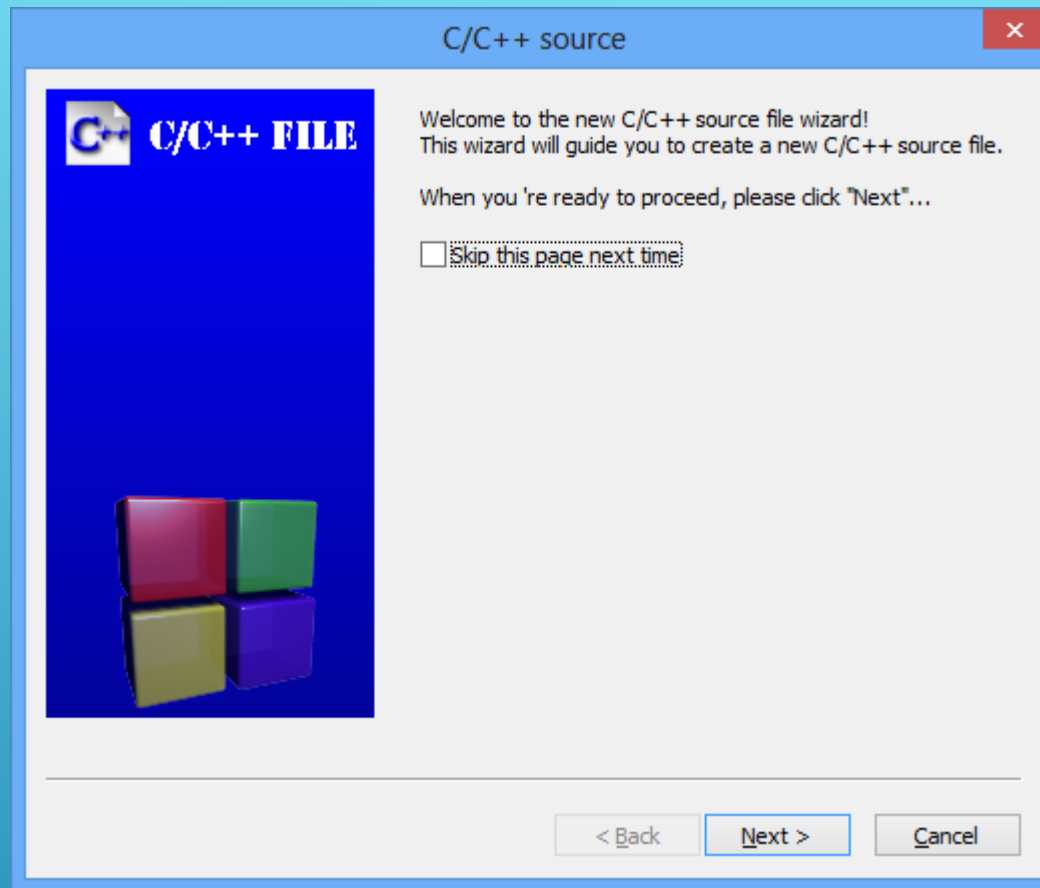
PROJETO #2 – HELLO WORLD ALLEGRO!

Several thin, parallel white lines of varying lengths and orientations are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

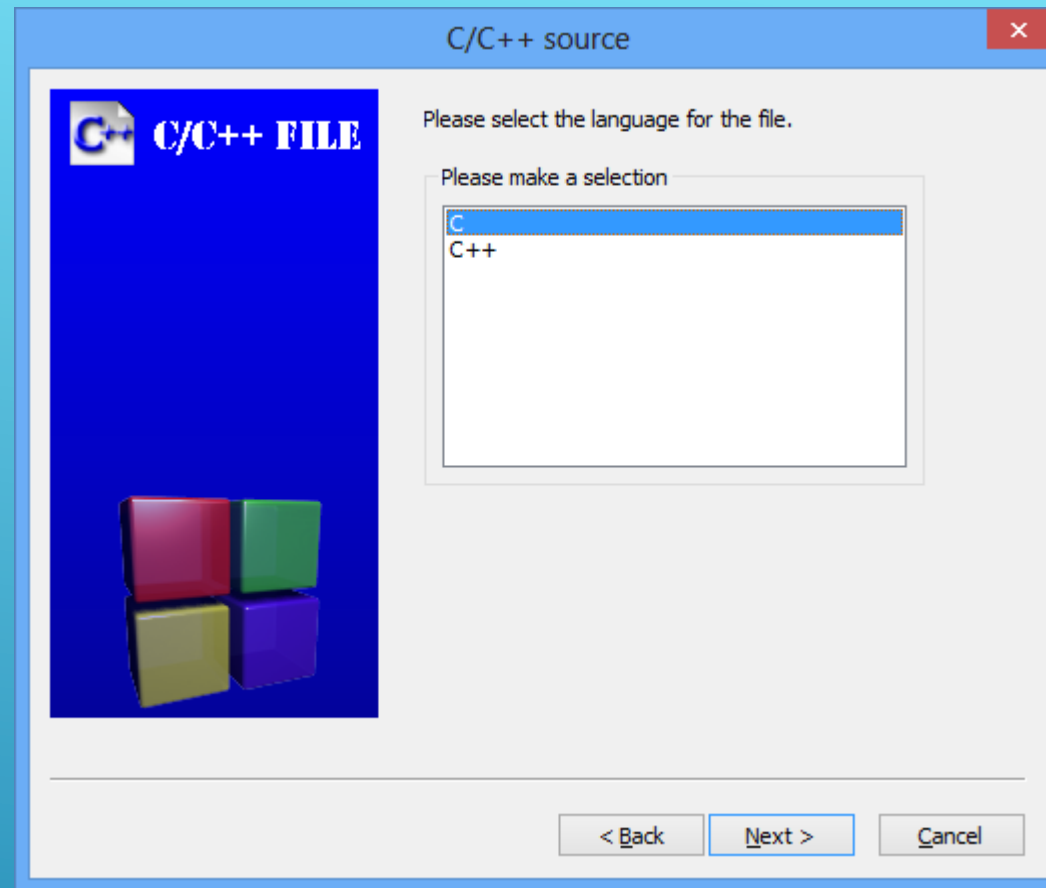




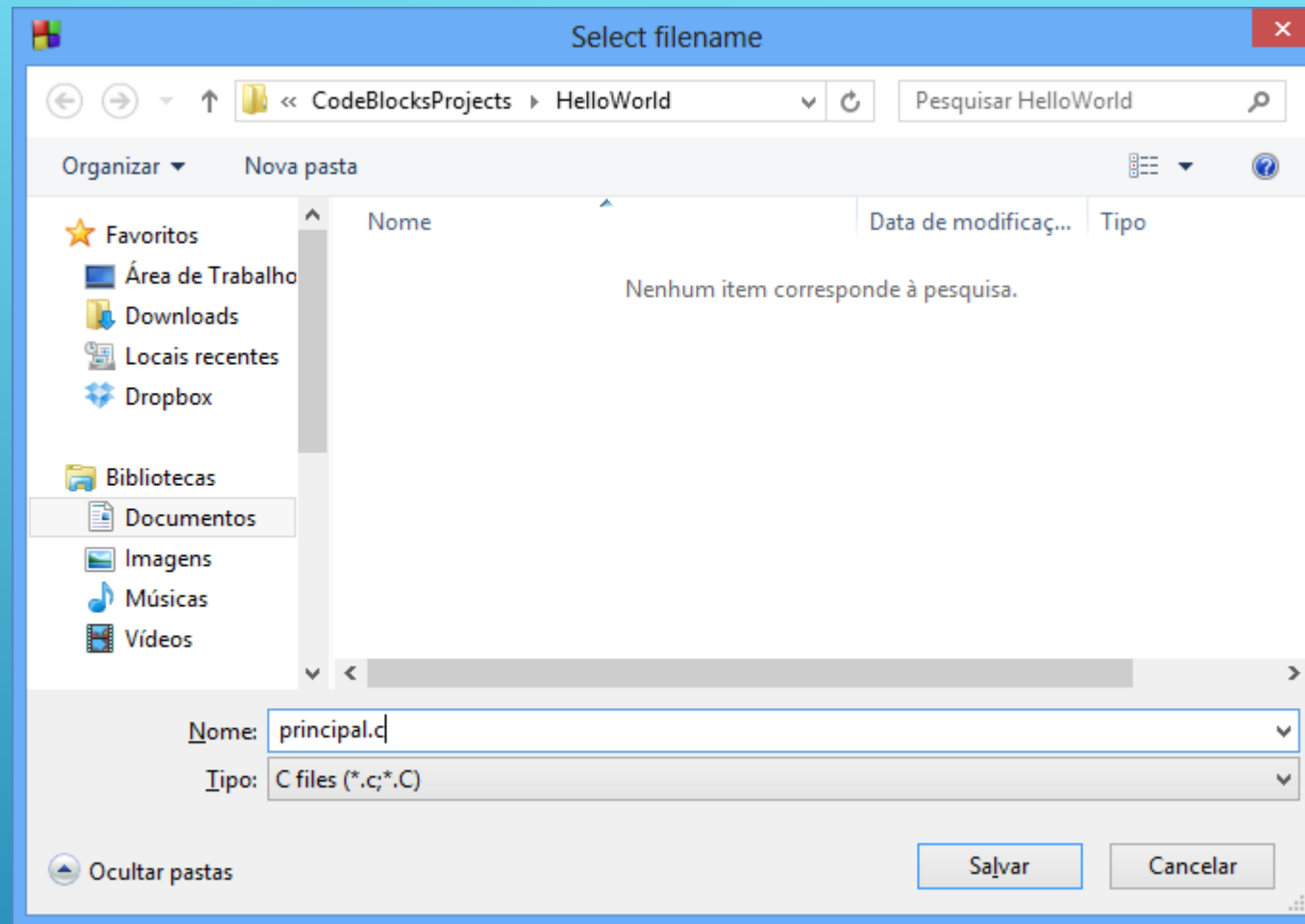
## PROJETO #2 – HELLO WORLD ALLEGRO!



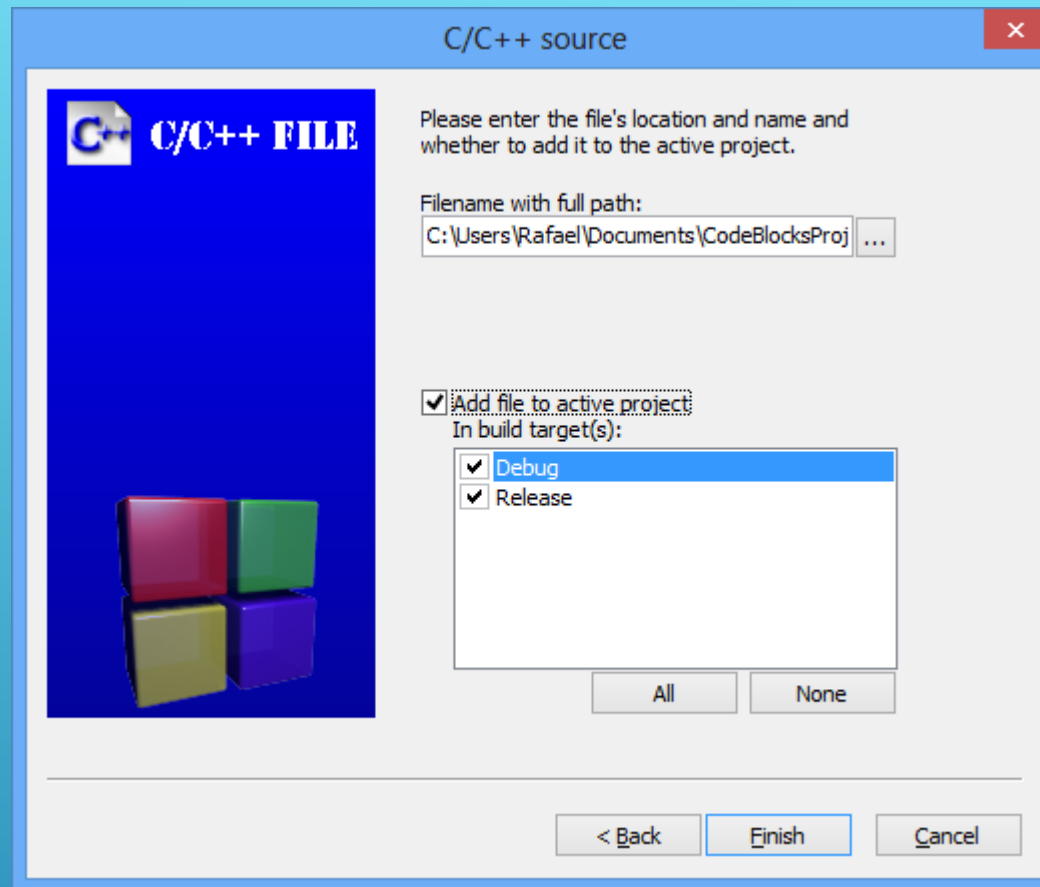
PROJETO #2 – HELLO WORLD ALLEGRO!



PROJETO #2 – HELLO WORLD ALLEGRO!



# PROJETO #2 – HELLO WORLD ALLEGRO!



## PROJETO #2 – HELLO WORLD ALLEGRO!


- ▶ Project -> Build Options...
- ▶ Aba Linker Settings
- ▶ Quadro Other linker options:  
`-lallegro-5.0.9-mt`

PROJETO #2 – HELLO WORLD ALLEGRO!

- ▶ **Project -> Properties...**
- ▶ **Aba Build targets**
- ▶ **Seção Selected build target options**
- ▶ **Opção Type**
  - ▶ **Selecionar GUI application**

PROJETO #2 – HELLO WORLD ALLEGRO!

```
int main()  
{  
    return 0;  
}
```



O básico de todo  
programa em C:  
a função `main()`





Incluimos a  
biblioteca Allegro

```
#include <allegro5/allegro.h>
```

```
int main()  
{  
    return 0;  
}
```

```
#include <allegro5/allegro.h>
```

```
int main()
```

```
{
```

```
    ALLEGRO_DISPLAY *janela = NULL;
```

```
    return 0;
```

```
}
```

Criamos uma  
variável para  
representar a janela

Inicializamos com  
um valor nulo:  
NULL

O “tipo” da janela  
se chama  
ALLEGRO\_DISPLAY

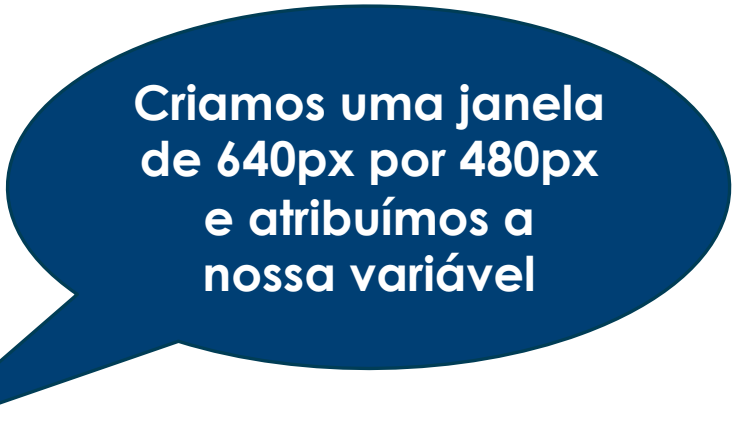
```
#include <allegro5/allegro.h>
```

```
int main()  
{  
    ALLEGRO_DISPLAY *janela = NULL;  
  
    al_init();  
  
    return 0;  
}
```

**Essa função estará  
em todos os nossos  
programas. Ela  
inicia a Allegro**

```
#include <allegro5/allegro.h>
```

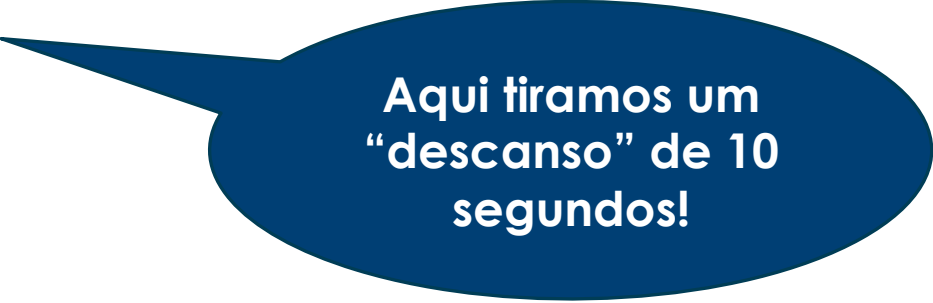
```
int main()  
{  
    ALLEGRO_DISPLAY *janela = NULL;  
  
    al_init();  
  
    janela = al_create_display(640, 480);  
  
    return 0;  
}
```



Criamos uma janela  
de 640px por 480px  
e atribuímos a  
nossa variável

```
#include <allegro5/allegro.h>
```

```
int main()  
{  
    ALLEGRO_DISPLAY *janela = NULL;  
  
    al_init();  
  
    janela = al_create_display();  
  
    al_rest(10.0);  
  
    return 0;  
}
```

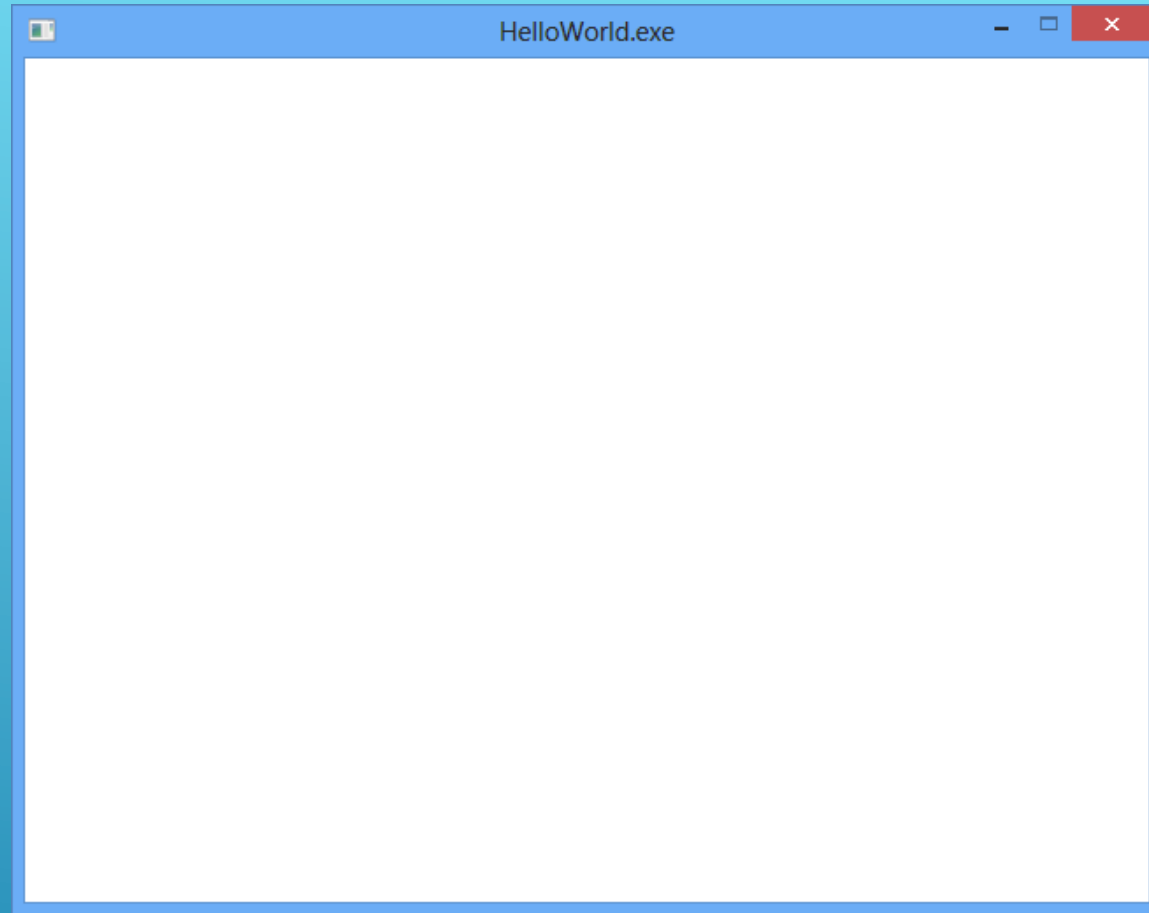


Aqui tiramos um  
“descanso” de 10  
segundos!

```
#include <allegro5/allegro.h>
```

```
int main()  
{  
    ALLEGRO_DISPLAY *janela = NULL;  
  
    al_init();  
  
    janela = al_create_display();  
  
    al_rest(10.0);  
  
    al_destroy_display(janela);  
  
    return 0;  
}
```

E por fim destruímos a  
janela que criamos  
para terminar



PROJETO #2 – HELLO WORLD ALLEGRO!

E VAMOS QUE VAMOS!

