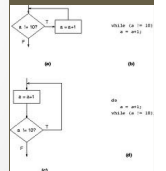
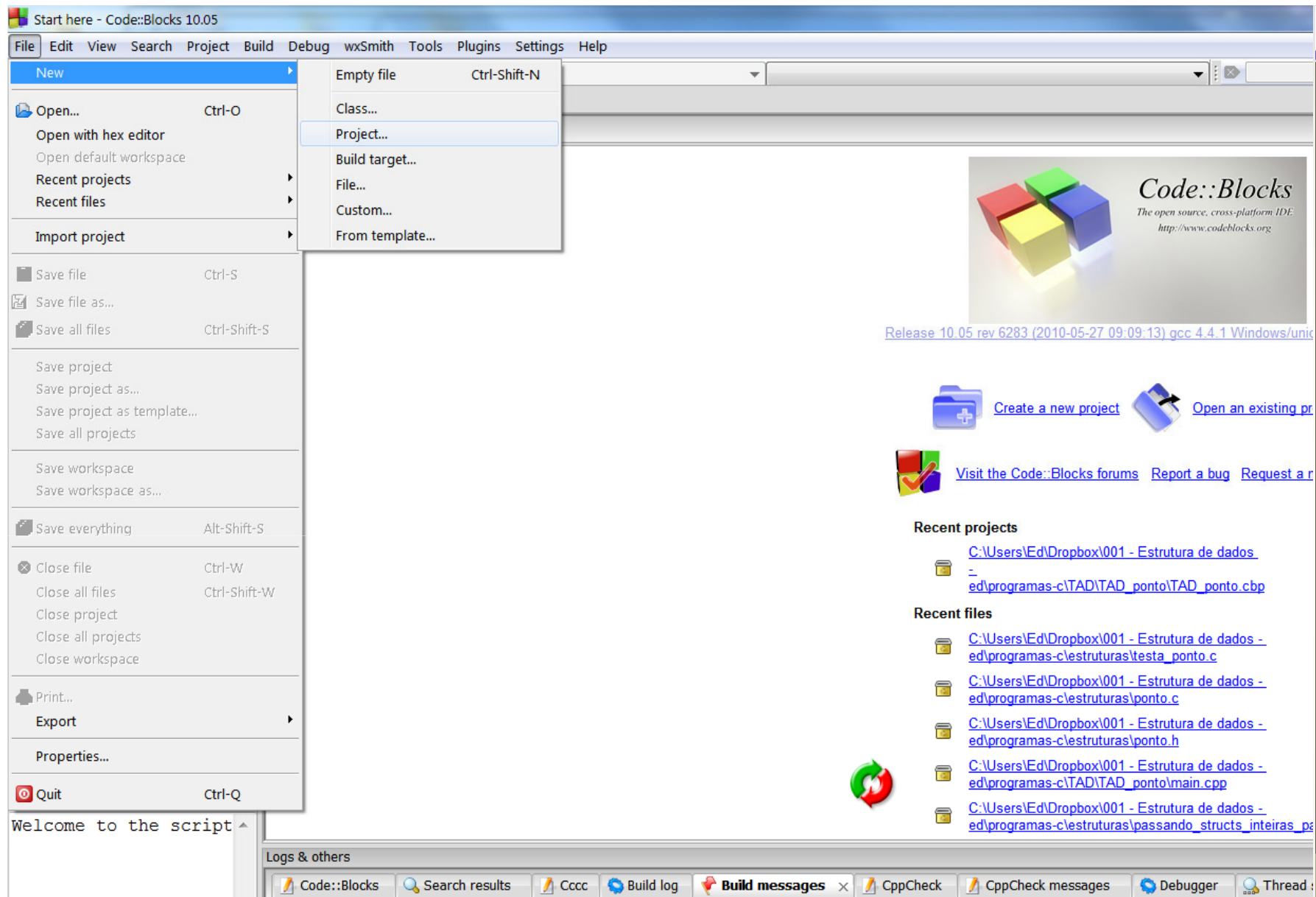


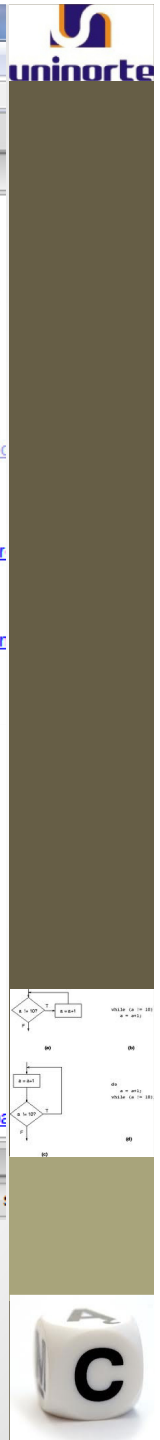
Tutorial de como compilar o TAD (criar arquivos cabeçalho .h)

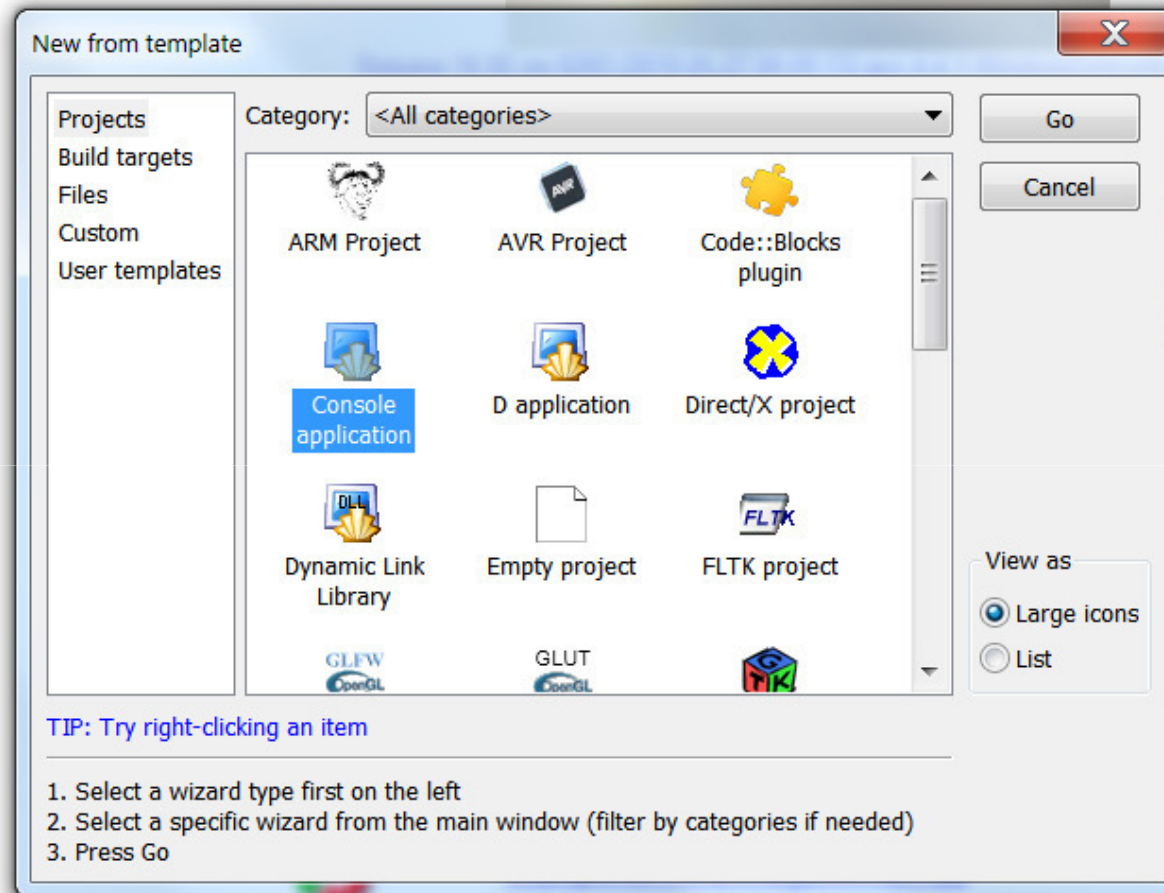
- Após os 3 arquivos estarem prontos (**ponto.h**, **ponto.c** e **testa_ponto.cpp** ou **testa_ponto.c**) você deve seguir os passos descritos nas imagens a seguir.
- As imagens se referem ao CodeBlocks.
- No DevC++ basta criar um projeto e adicionar ao mesmo os três arquivos
- (Ao criar o projeto siga “File” > “New” > “Project” > “Console Application”.
- Na esquerda da tela, onde aparece o nome do seu projeto, exclua o arquivo "main.c" criado pela IDE e adicione os seus arquivos)





File → New → Project



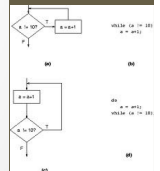


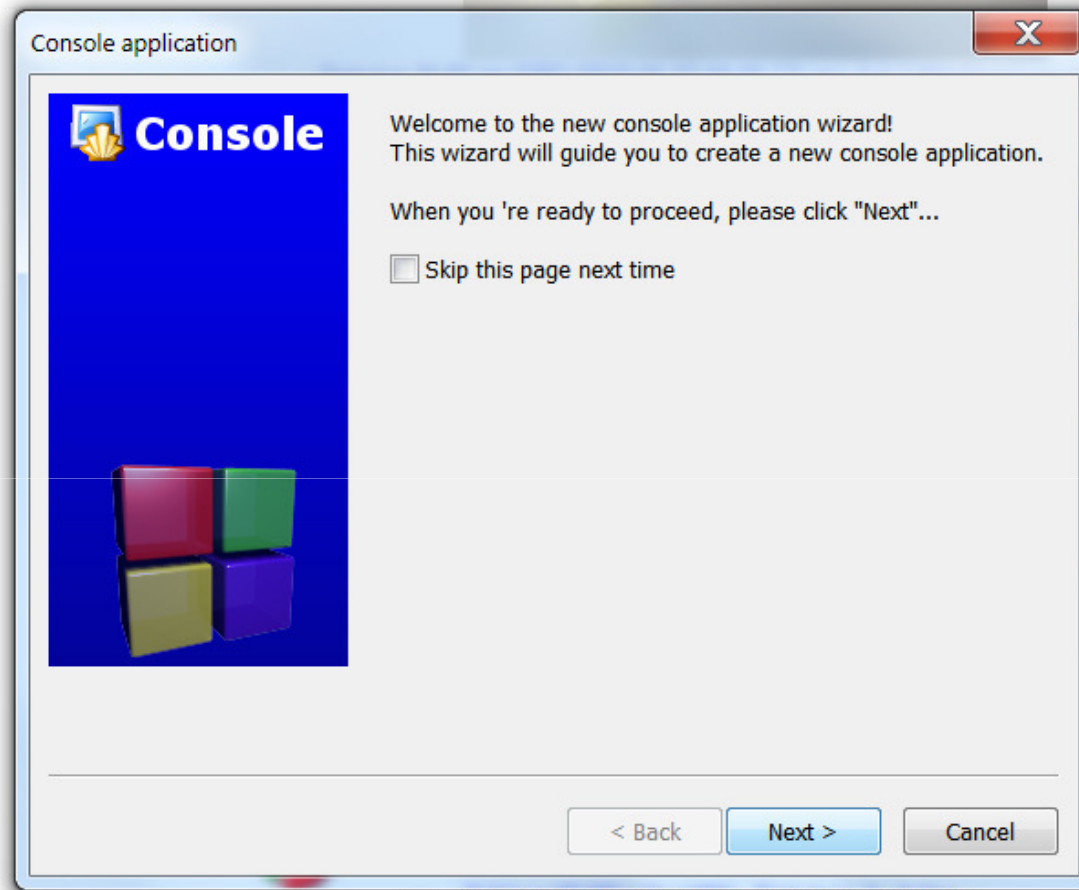
32 bit

ature

C:\Users\Ed\Dropbox\001 - Estrutura de dados - ed\programas-c\estruturas\passando_structs_inteiras_para_funcoes.cpp

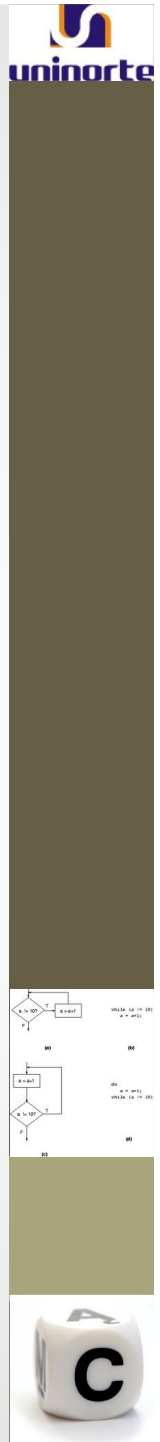
Escolha “Console Application”

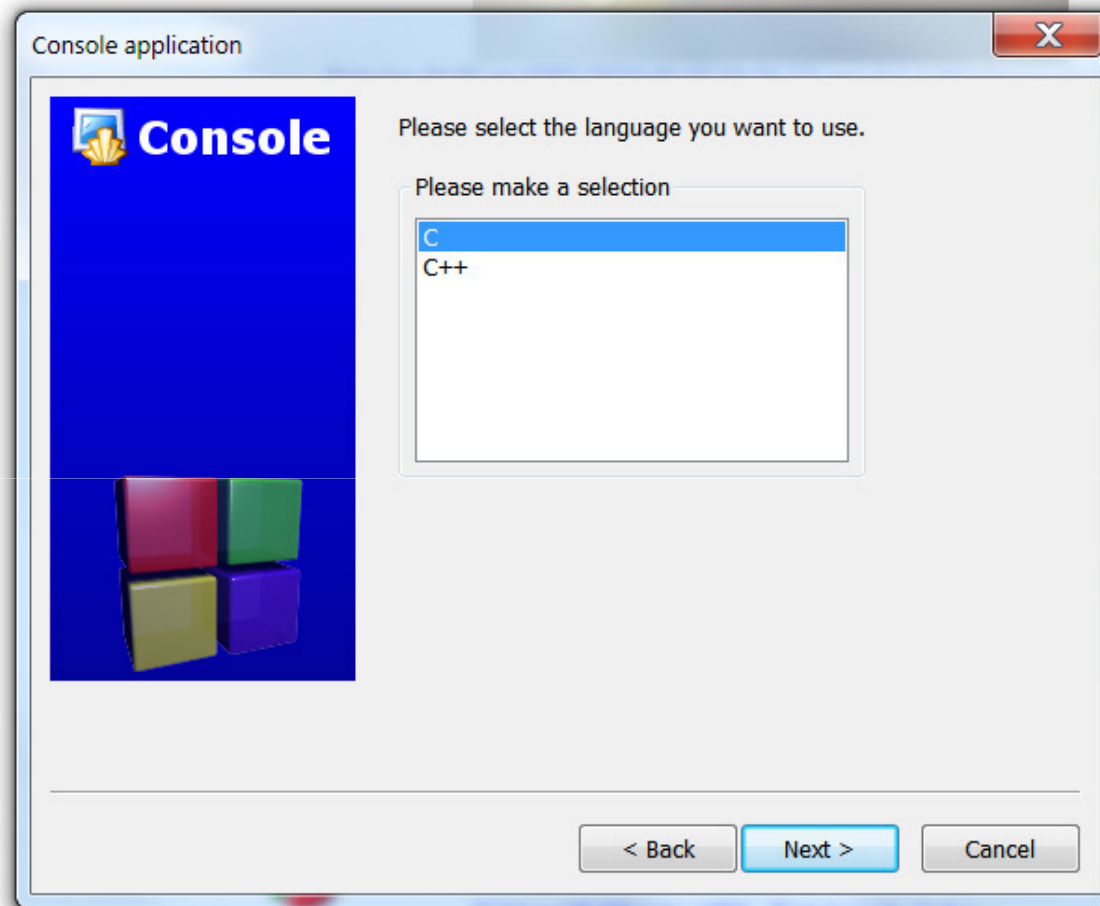




C:\Users\Ed\Dropbox\001 - Estrutura de dados -
ed\programas-c\estruturas\passando_structs_inteiros_para_funcoes.cpp

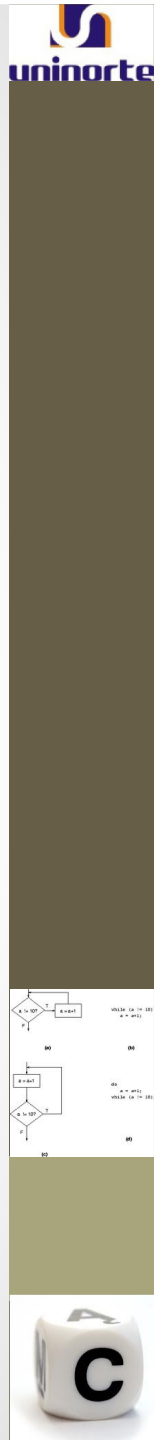
Clique em “next”

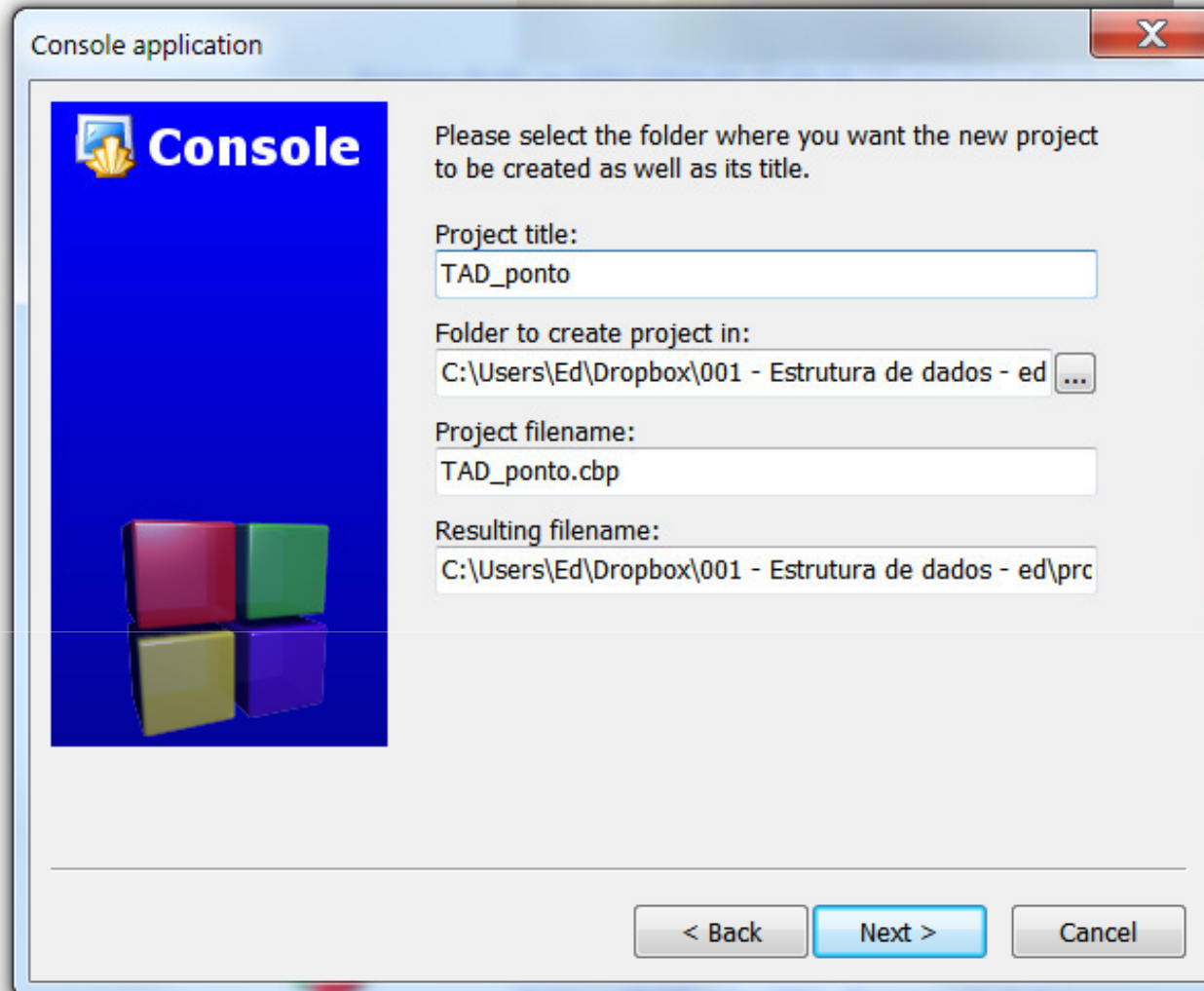




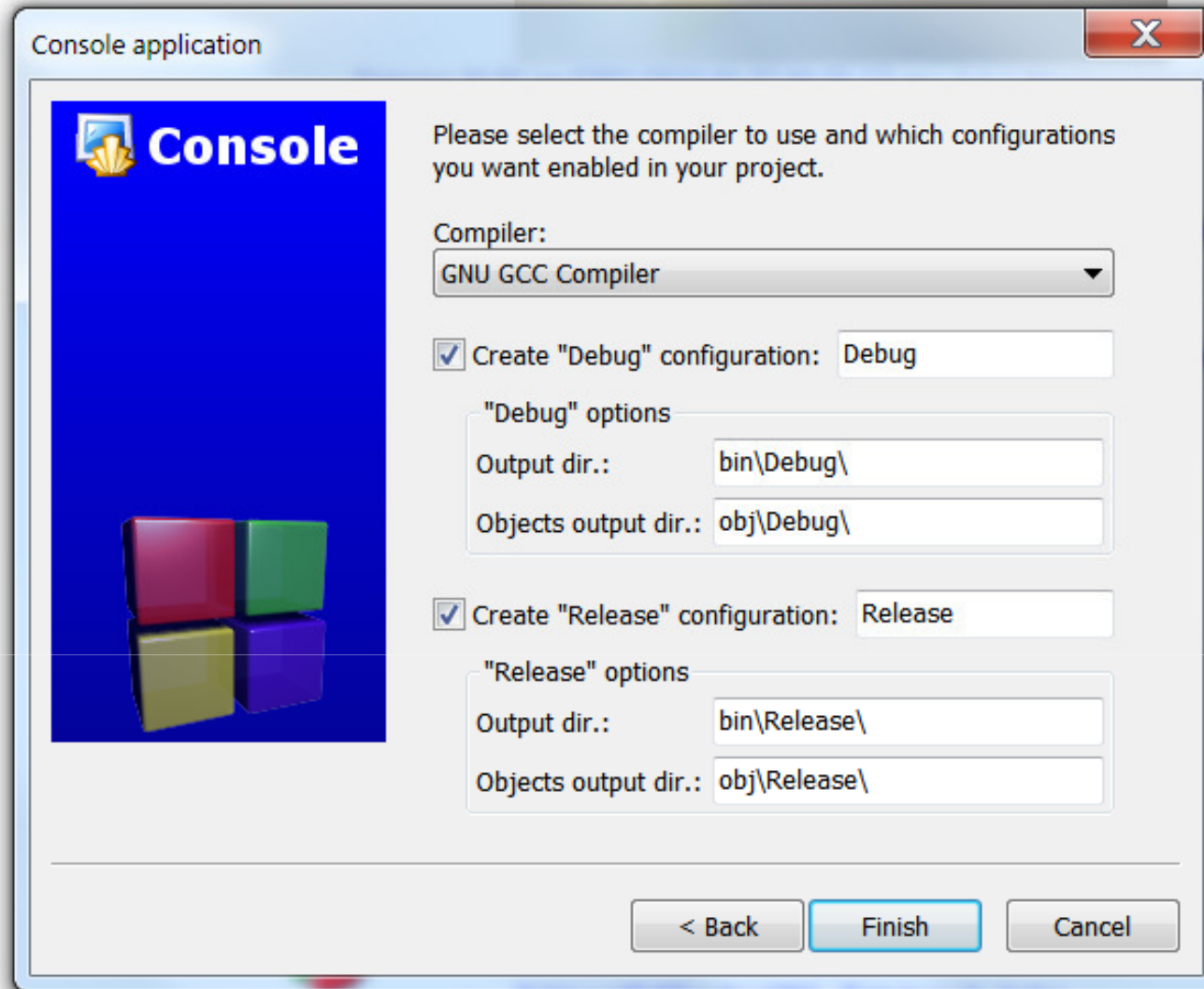
[C:\Users\Ed\Dropbox\001 - Estrutura de dados - ed\programas-c\estruturas\passando_structs_inteiros_para_funcoes.cpp](#)

Escolha “C” (se salvou .c ou C++ se salvou .cpp)

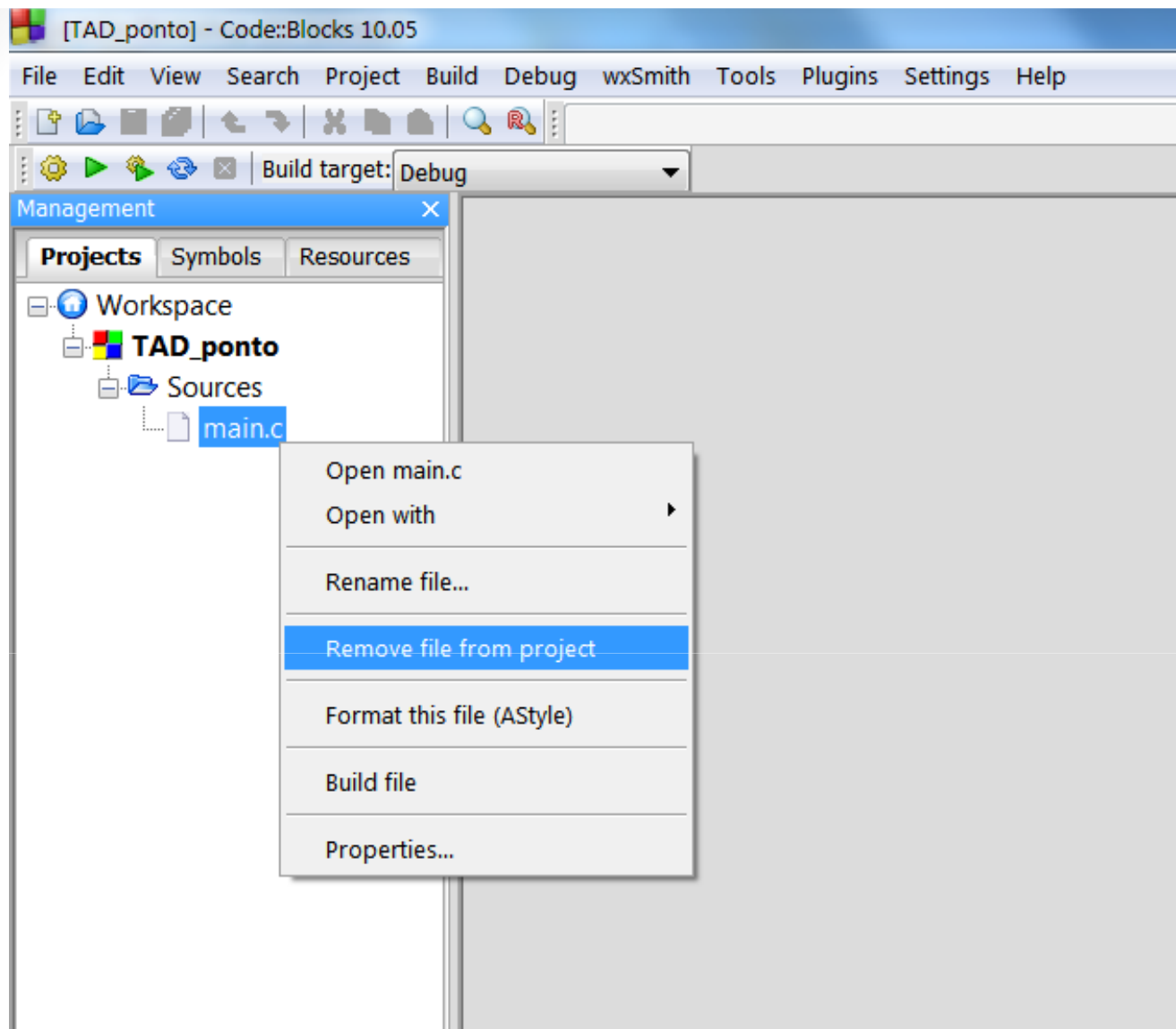




**Preencha com os dados adequados
(apenas o “project title”)**

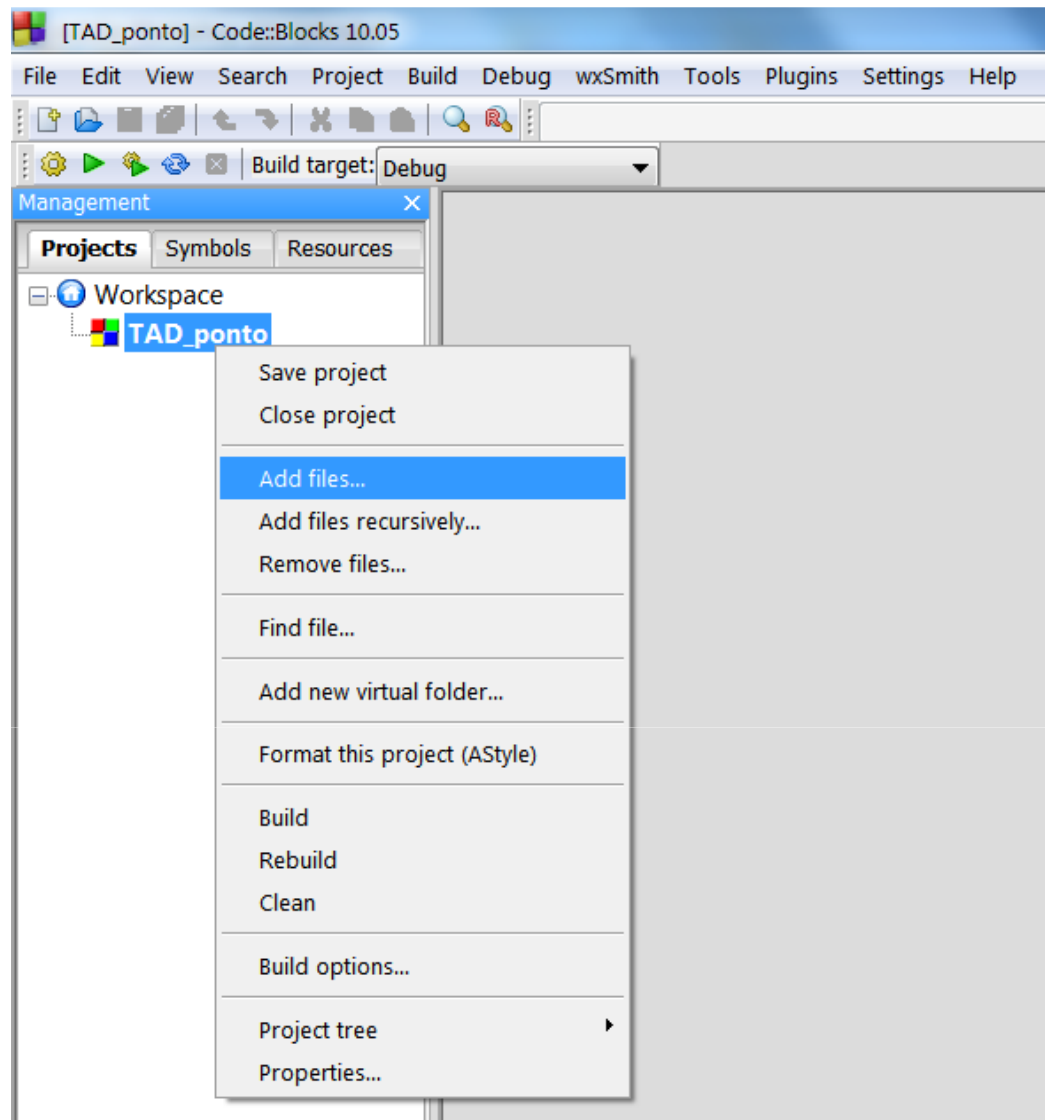


Escolha seu compilador instalado (não é necessário alterar outras opções)

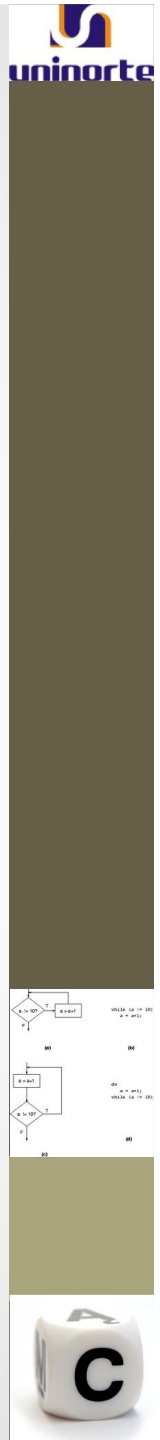


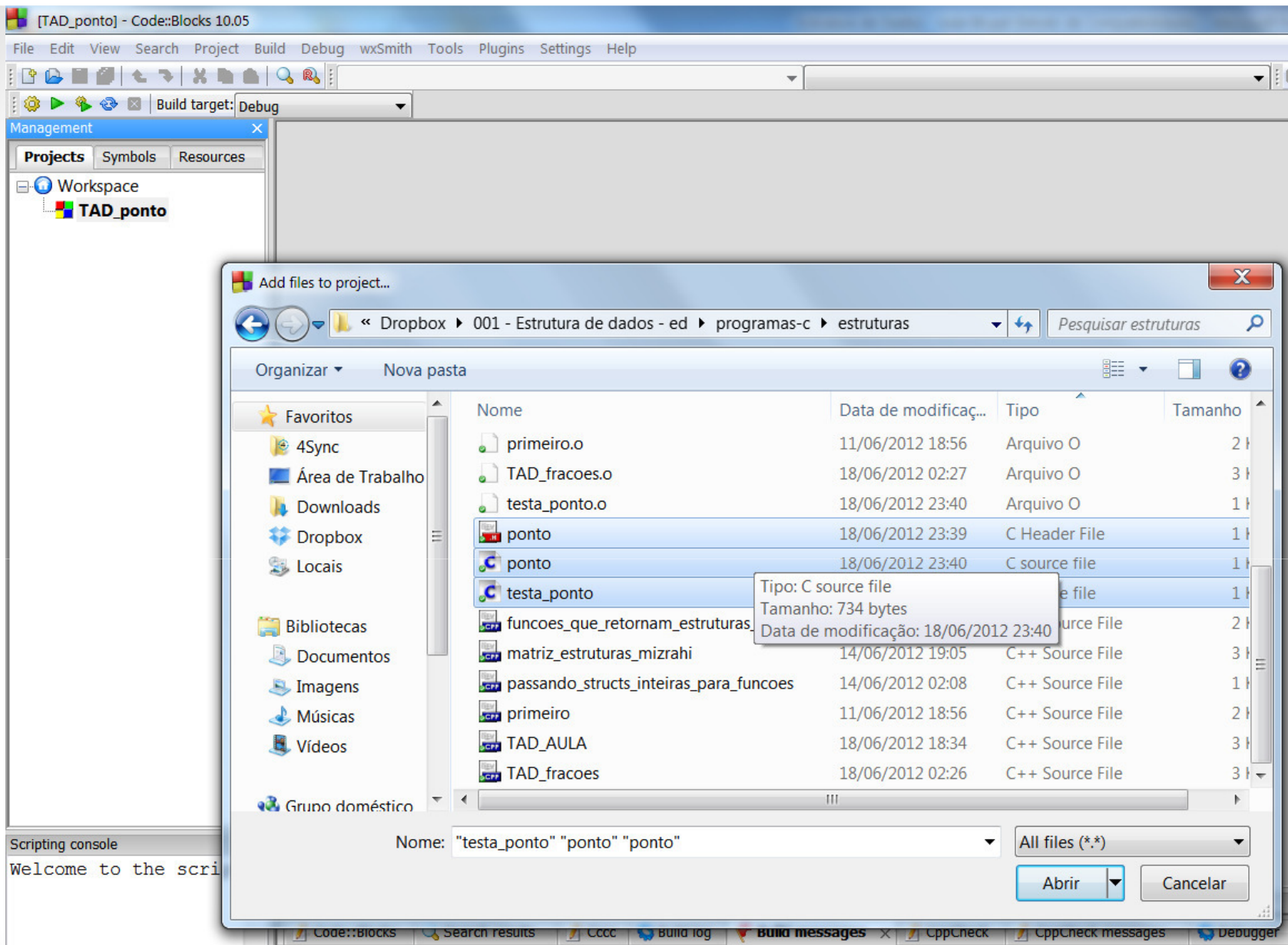
Remova o arquivo criado pelo programa automaticamente (clique com o botão direito)





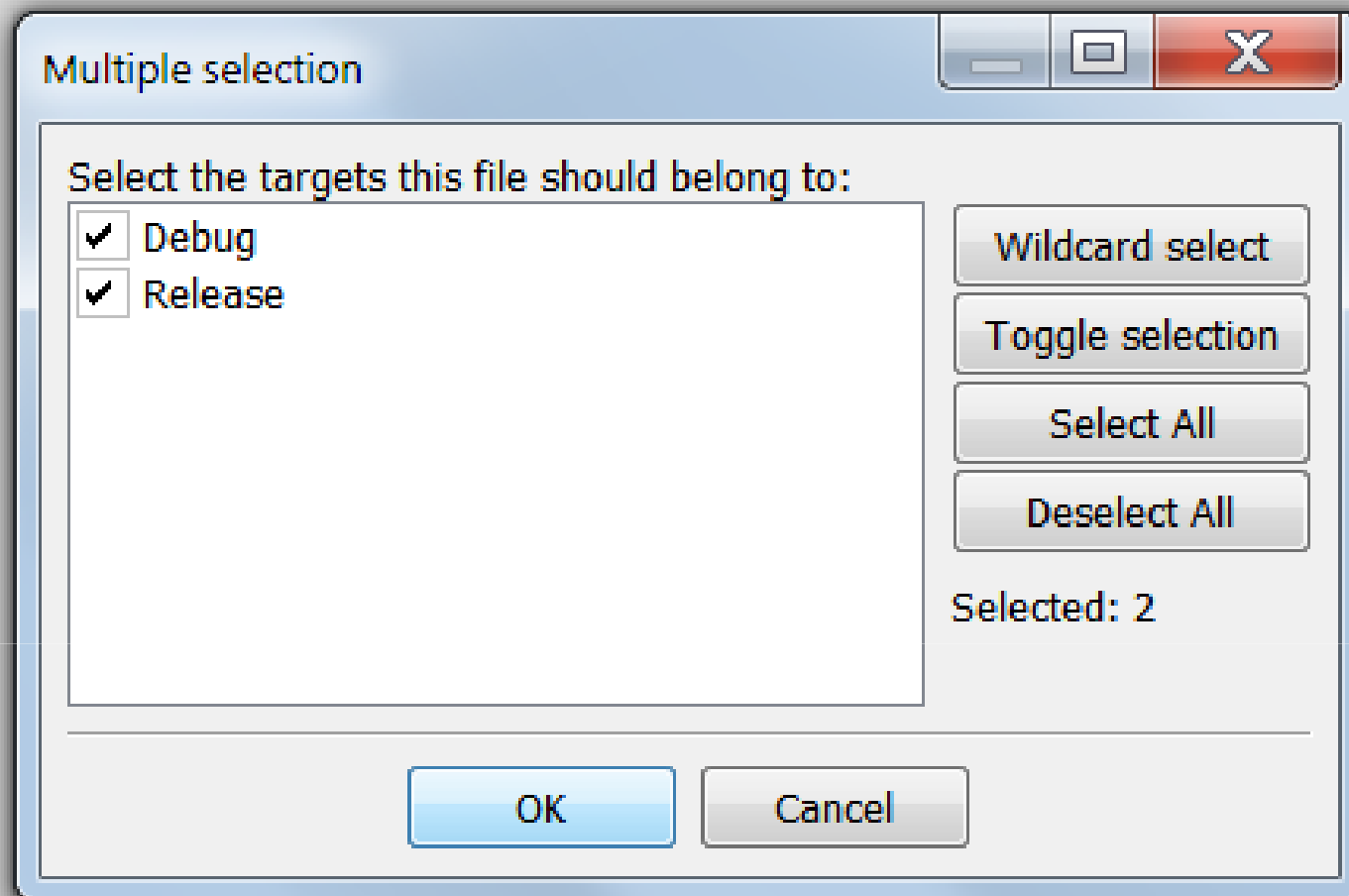
**Clique com o botão direito sobre seu projeto e
selecione “Add Files” (para adicionar seus arquivos
– ponto.c, ponto.h e testa_ponto.c)**



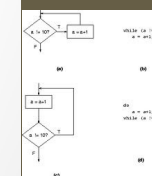


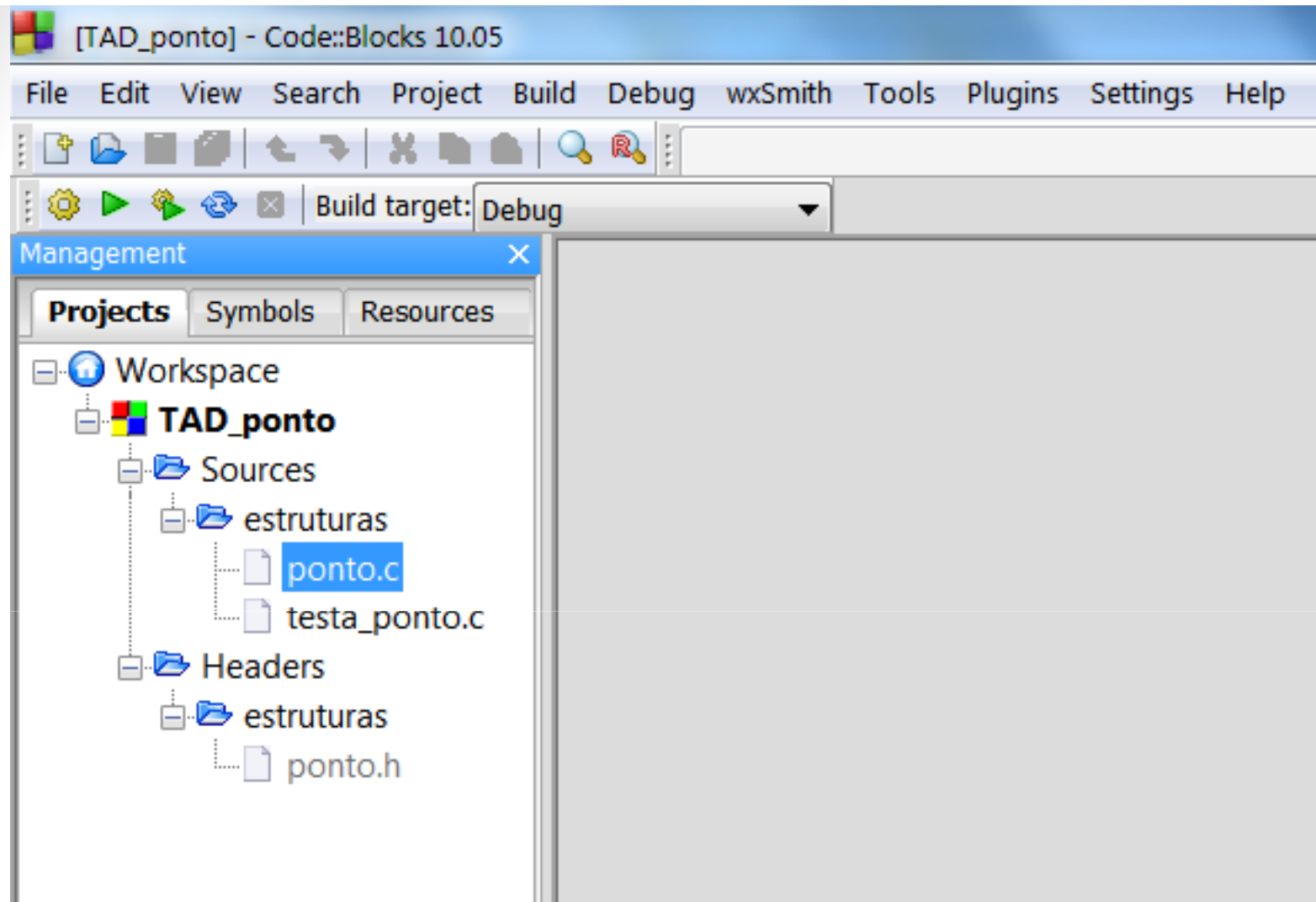
Localize e adicione seus arquivos





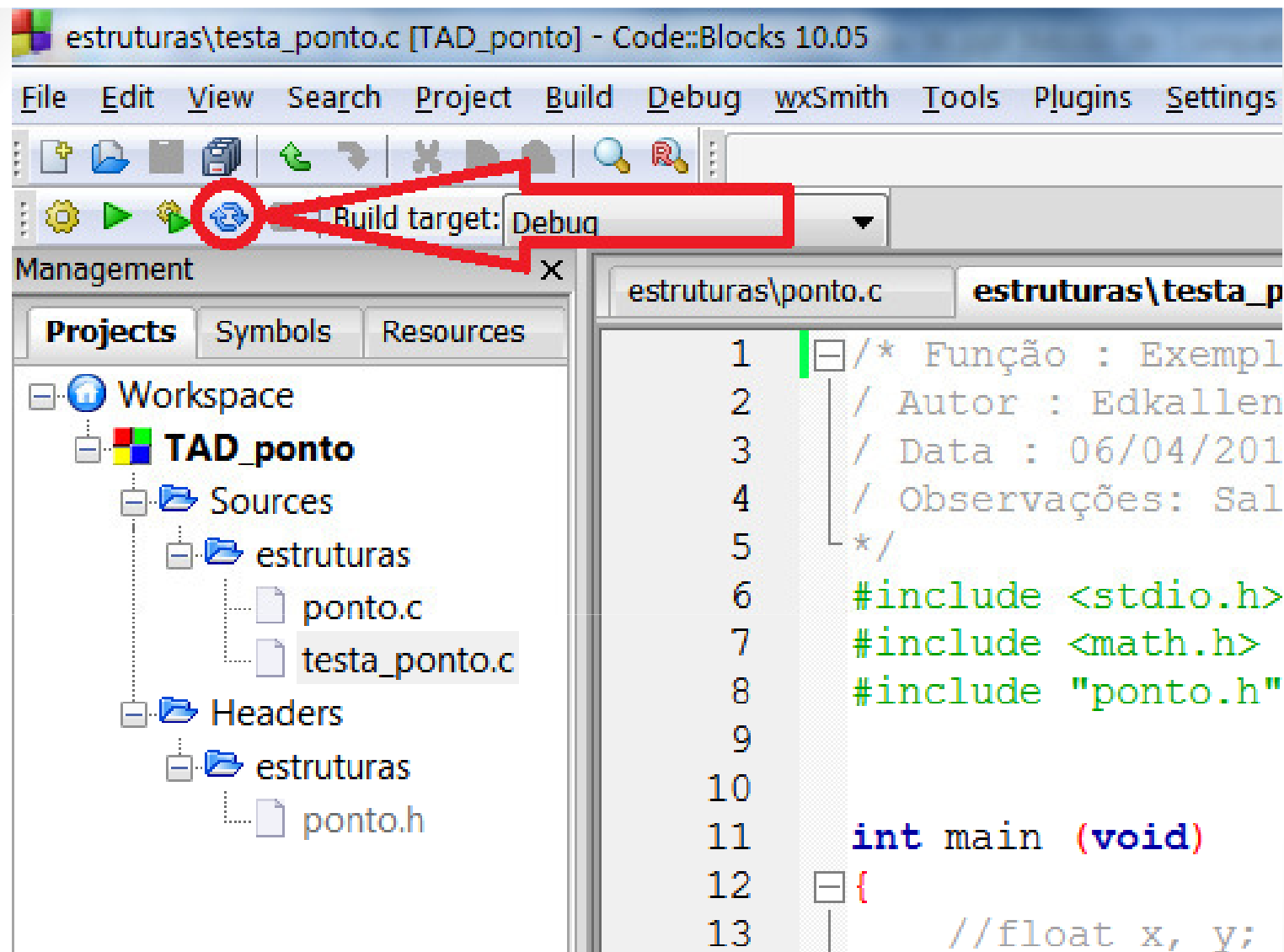
Clique “OK” na janela que aparece



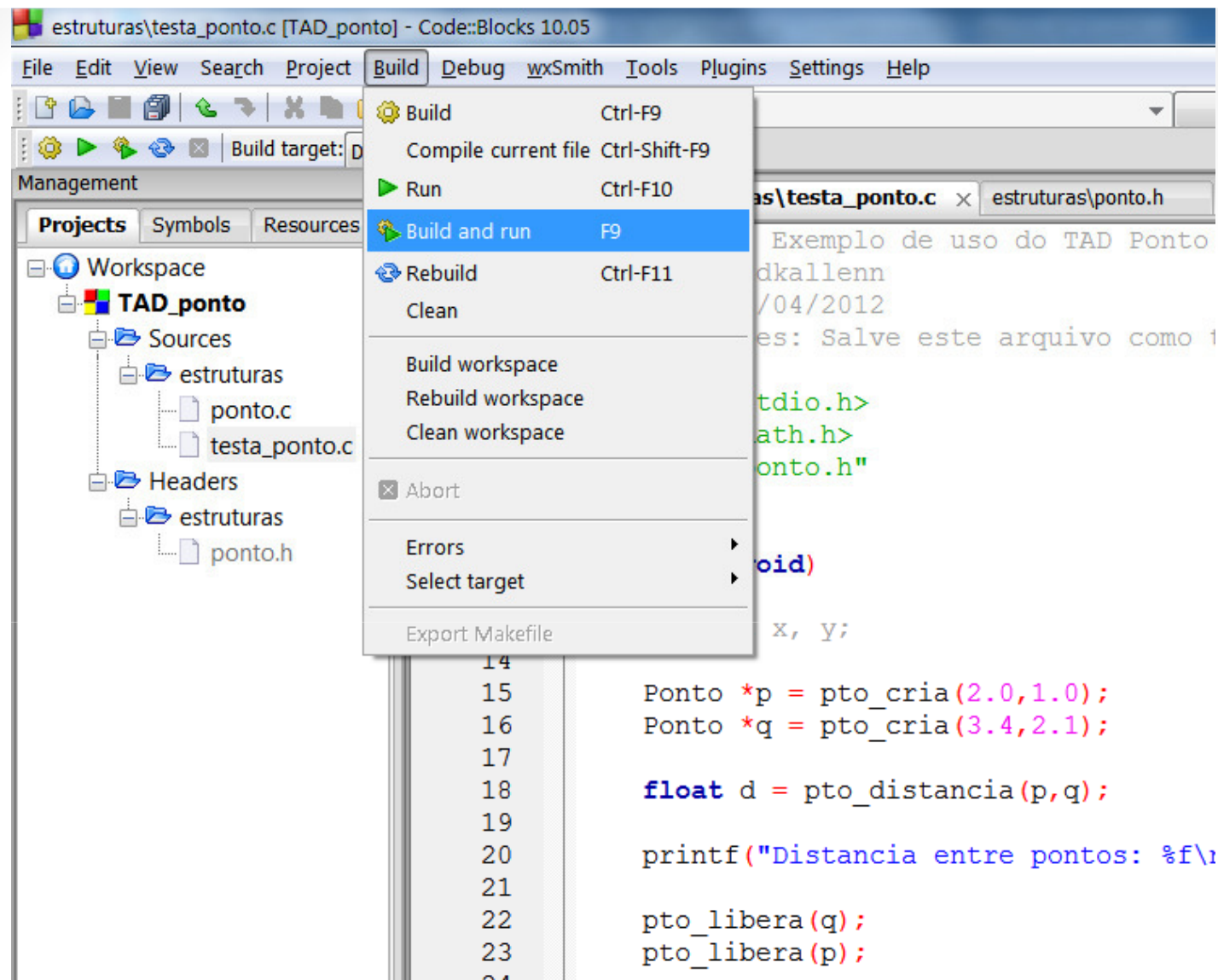


**Seus arquivos serão adicionados
ao projeto**





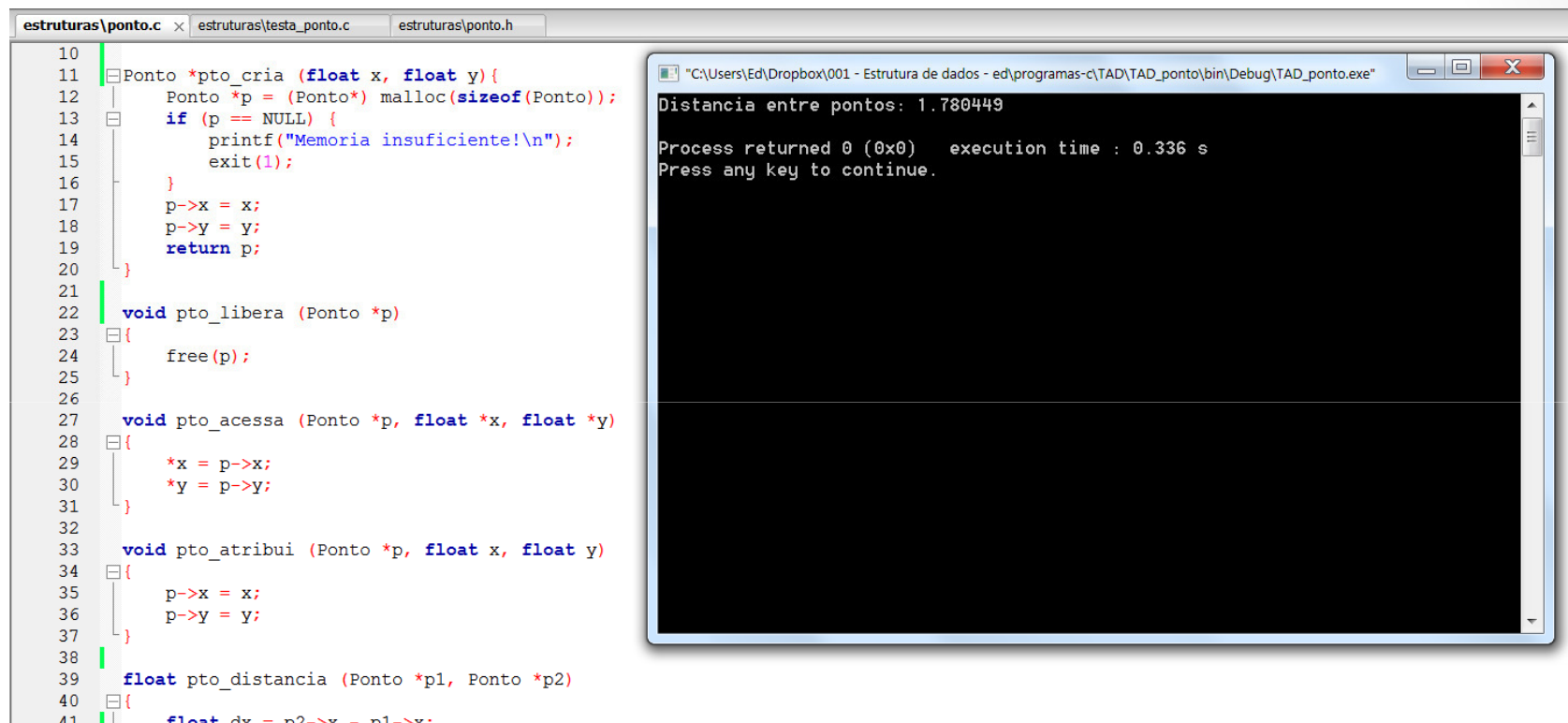
Abra-os (clique duas vezes sobre todos)
Clique em Rebuild. Depois, confirme.



Depois clique em “Build” → “Build and Run”



Eis o resultado:



The image shows a code editor with three tabs: `estruturas\ponto.c`, `estruturas\testa_ponto.c`, and `estruturas\ponto.h`. The `ponto.c` file contains the following code:

```

10
11 Ponto *pto_cria (float x, float y){
12     Ponto *p = (Ponto*) malloc(sizeof(Ponto));
13     if (p == NULL) {
14         printf("Memoria insuficiente!\n");
15         exit(1);
16     }
17     p->x = x;
18     p->y = y;
19     return p;
20 }
21
22 void pto_libera (Ponto *p)
23 {
24     free(p);
25 }
26
27 void pto_acessa (Ponto *p, float *x, float *y)
28 {
29     *x = p->x;
30     *y = p->y;
31 }
32
33 void pto_atribui (Ponto *p, float x, float y)
34 {
35     p->x = x;
36     p->y = y;
37 }
38
39 float pto_distancia (Ponto *p1, Ponto *p2)
40 {
41     float dx = p2->x - p1->x;

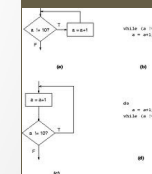
```

Overlaid on the code editor is a console window titled `"C:\Users\Ed\Dropbox\001 - Estrutura de dados - ed\programas-c\TAD\TAD_ponto\bin\Debug\TAD_ponto.exe"`. The console output is:

```

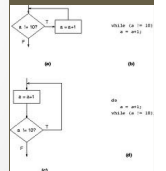
Distancia entre pontos: 1.780449
Process returned 0 (0x0)   execution time : 0.336 s
Press any key to continue.

```



RESUMO

- **Módulo** → arquivo com funções que representam apenas parte da implementação de um programa completo
- **Arquivo objeto** → resultado de compilar um módulo geralmente com extensão .o ou .obj
- **Interface de módulo** → arquivo contendo apenas os protótipos das funções oferecidas pelo módulo, e os tipos de dados exportados pelo módulo
- **TAD** → define um novo tipo de dado e o conjunto de operações para manipular dados do tipo



**VER A LISTA DE
EXERCÍCIOS QUE ESTARÁ
DISPONÍVEL NO BLOG E NO
DROPBOX.**

