# PROGRAMAÇÃO COMPUTACIONAL PARA ENGENHARIA

## INTEGRATED DEVELOPMENT ENVIRONMENT

Maurício Moreira Neto[1]

[1]**Universidade Federal do Ceará**
**Departamento de Computação**

30 de janeiro de 2020

# Sumário



DEPARTAMENTO DE COMPUTAÇÃO

# Objetivos

- Aprender sobre o que é uma IDE
- Aprender a utilizar uma IDE para criar e executar códigos na linguagem C
- Aprender quais são os recursos básicos de uma IDE para programação

DEPARTAMENTO DE COMPUTAÇÃO

# Definição

- IDE significa Ambiente de Desenvolvimento Integrado
  - Do inglês – Integrated Development Environment

- As IDEs são utilizadas para criar programas em diversas linguagens de programação, contendo ferramentas necessárias para o desenvolvimento do software

DEPARTAMENTO DE COMPUTAÇÃO

# Definição

- Existem IDEs específicas para cada tipo de linguagem
  - Também existem muitas IDEs que suportam o desenvolvimento de programas de múltiplas linguagens de programação

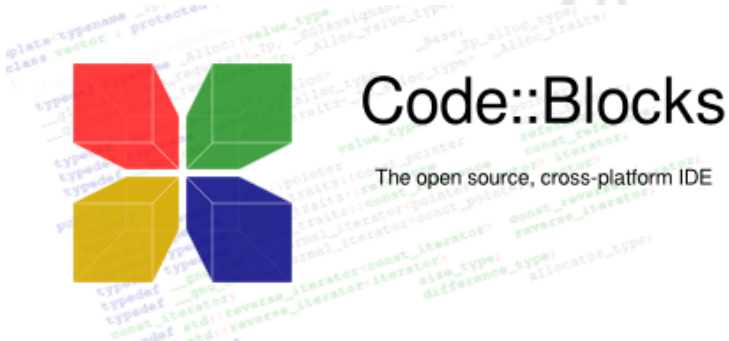
- Uma das mais famosas IDEs utilizada para desenvolver programas com a linguagem C é o **Code::Blocks**

DEPARTAMENTO DE COMPUTAÇÃO

# Code Blocks

- O Code::Blocks pode ser baixado diretamente pelo seu site: `www.codeblocks.org`

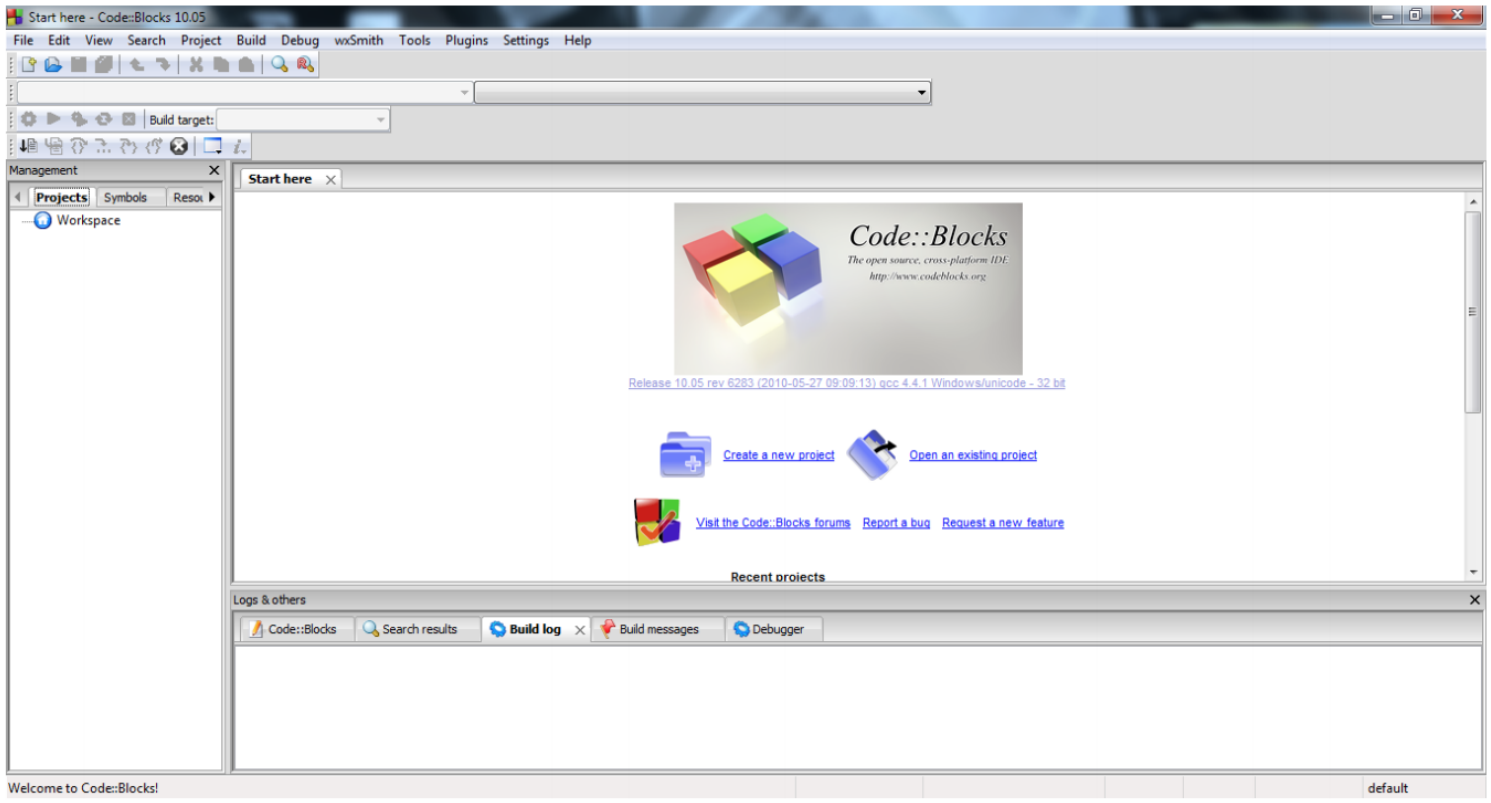
- Procure baixar a versão que inclui tanto a **IDE do Code::Blocks** como o **compilador GCC** e o **debugger GDB da MinGW**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

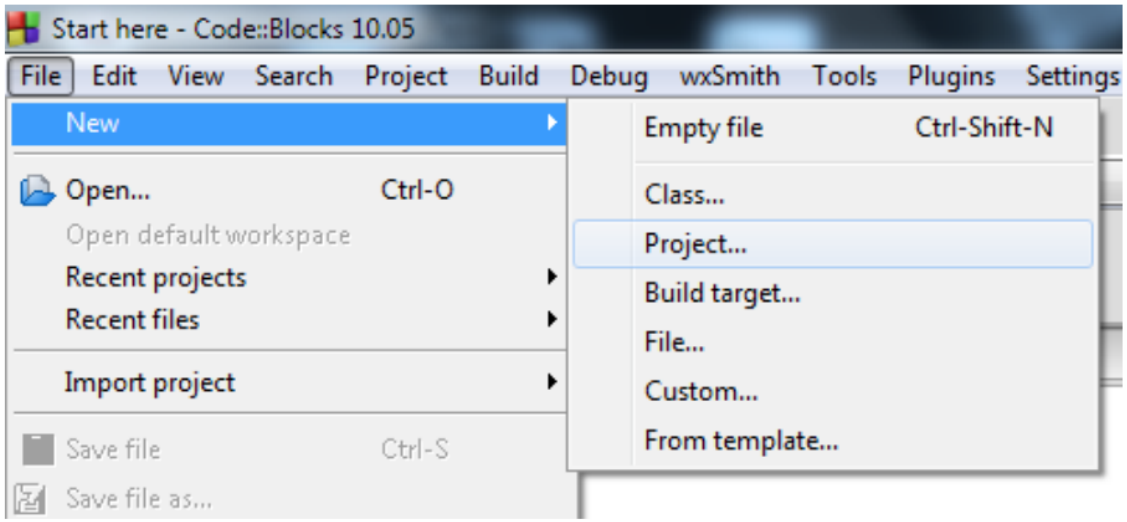
- Abra o software **Code::Blocks**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

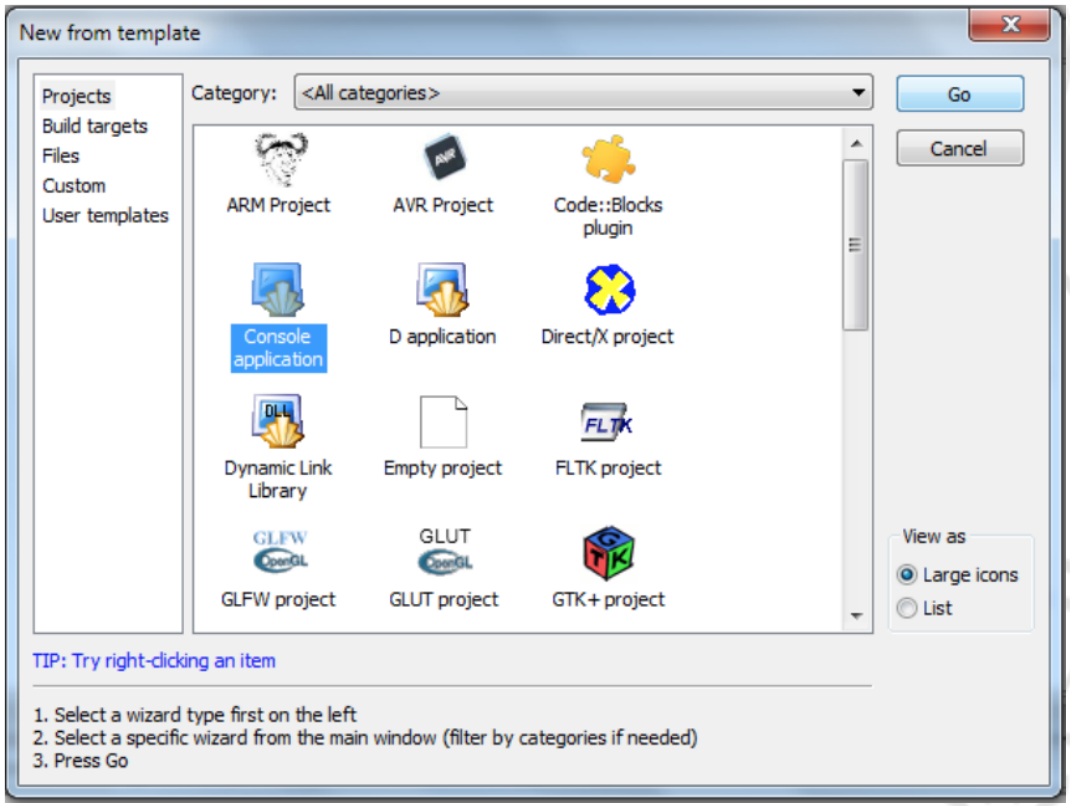
- Clique em **File**, escolha **New** e depois **Project**...




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

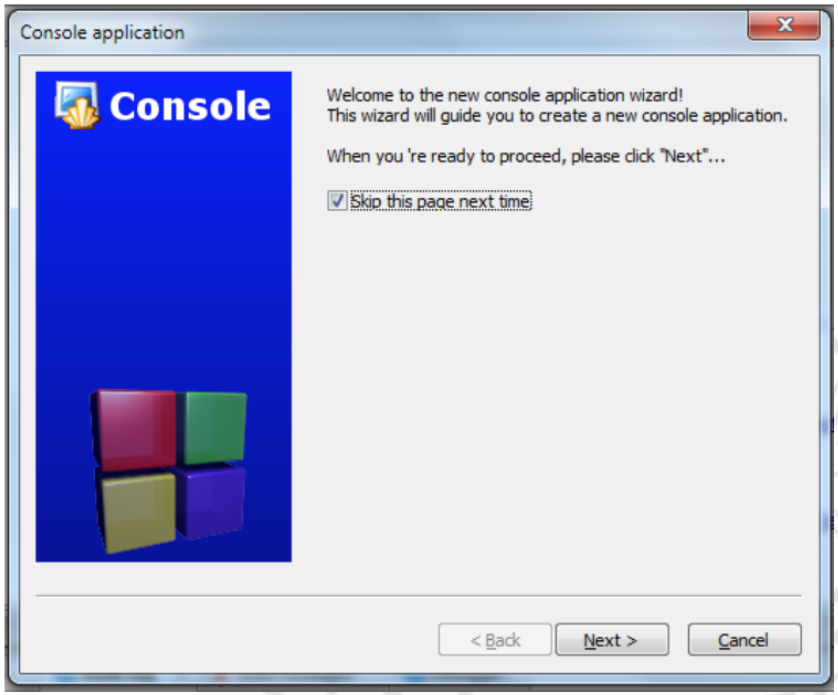
- Uma lista de modelos (templates) de projetos vai aparecer. Escolha **Console Application**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

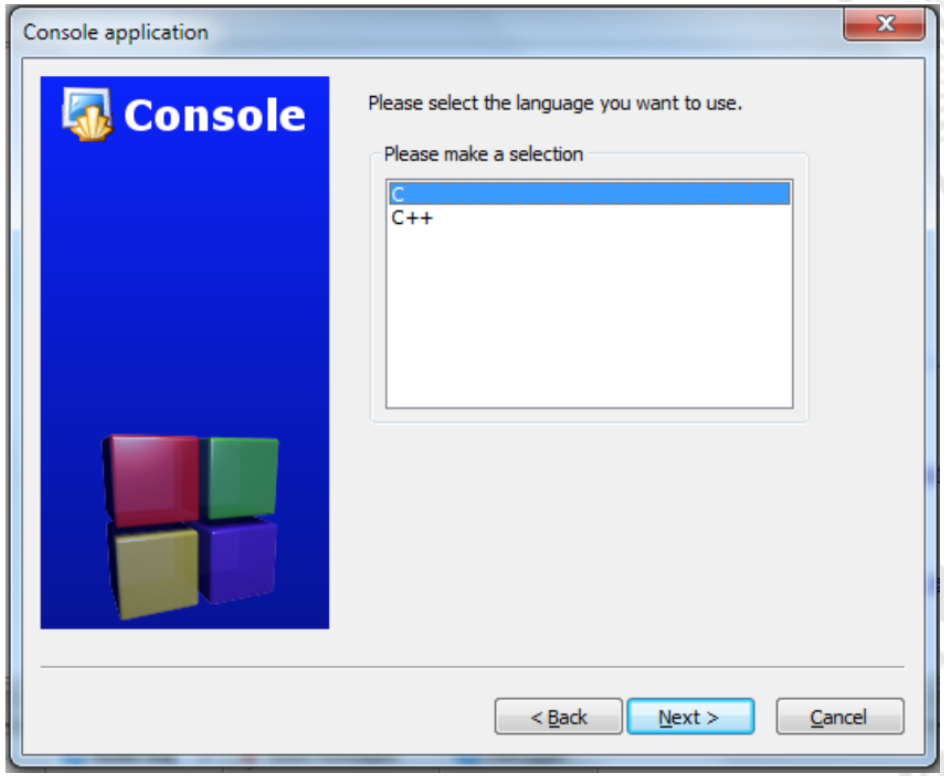
- Caso esteja criando um projeto pela primeira vez, a tela a seguir irá aparecer
- Em seguida, clique em **Next**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

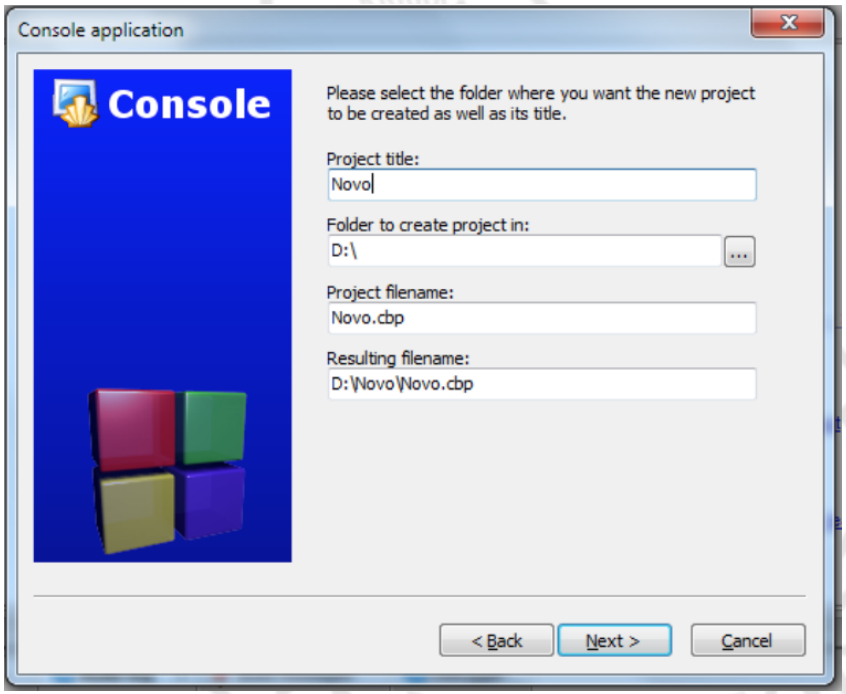
- Escolha a opção **C** e clique em **Next**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

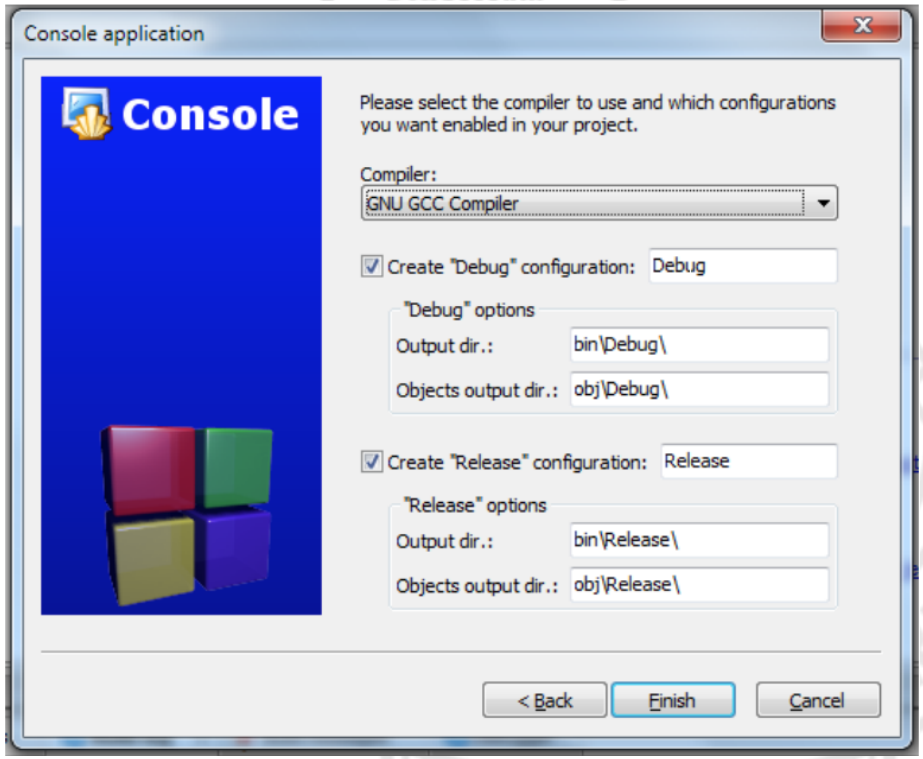
- **Project Title**: título do projeto
- **Folder to create project in**: caminho aonde o projeto será salvo
  - Não pode haver acentos ou espaços
- Ao final, clique em **Next**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

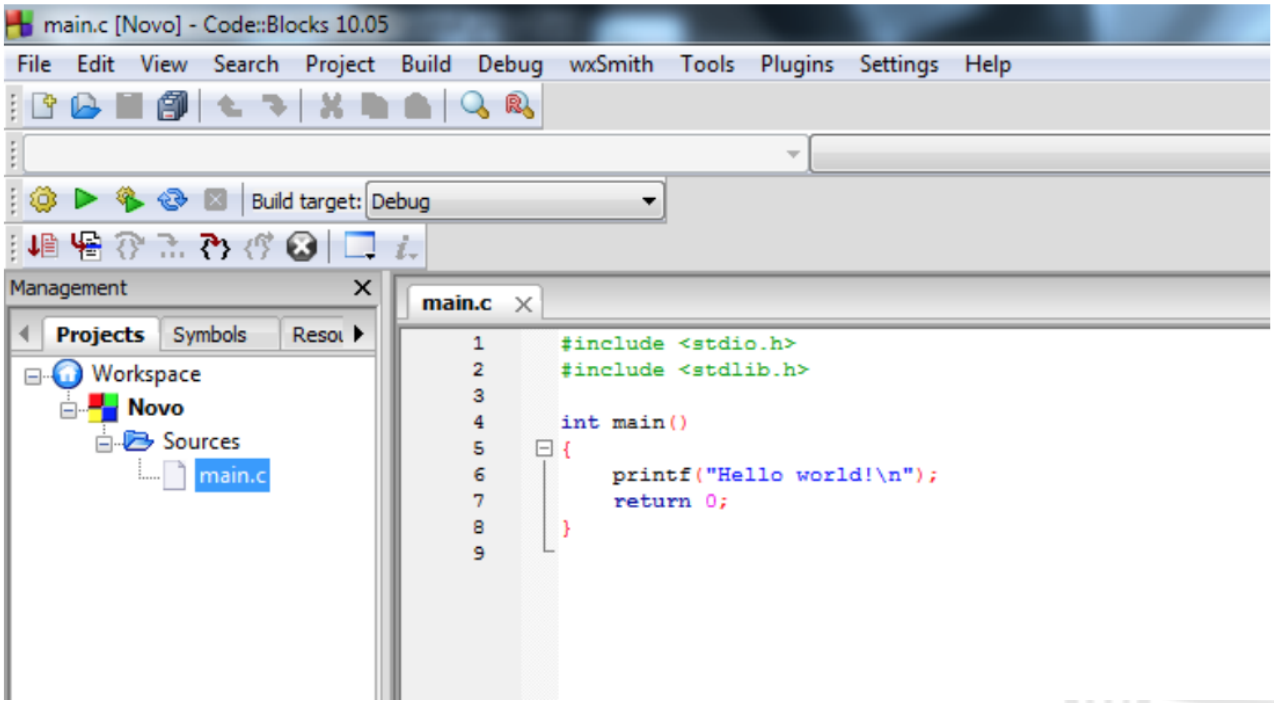
- Na tela a seguir, algumas configurações do compilador que podem ser modificadas
  - No entanto, isso não será necessário!
  - Clique em **Finish**




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

- Ao fim destes passos, é criado um projeto com um código de “Hello World” simples.



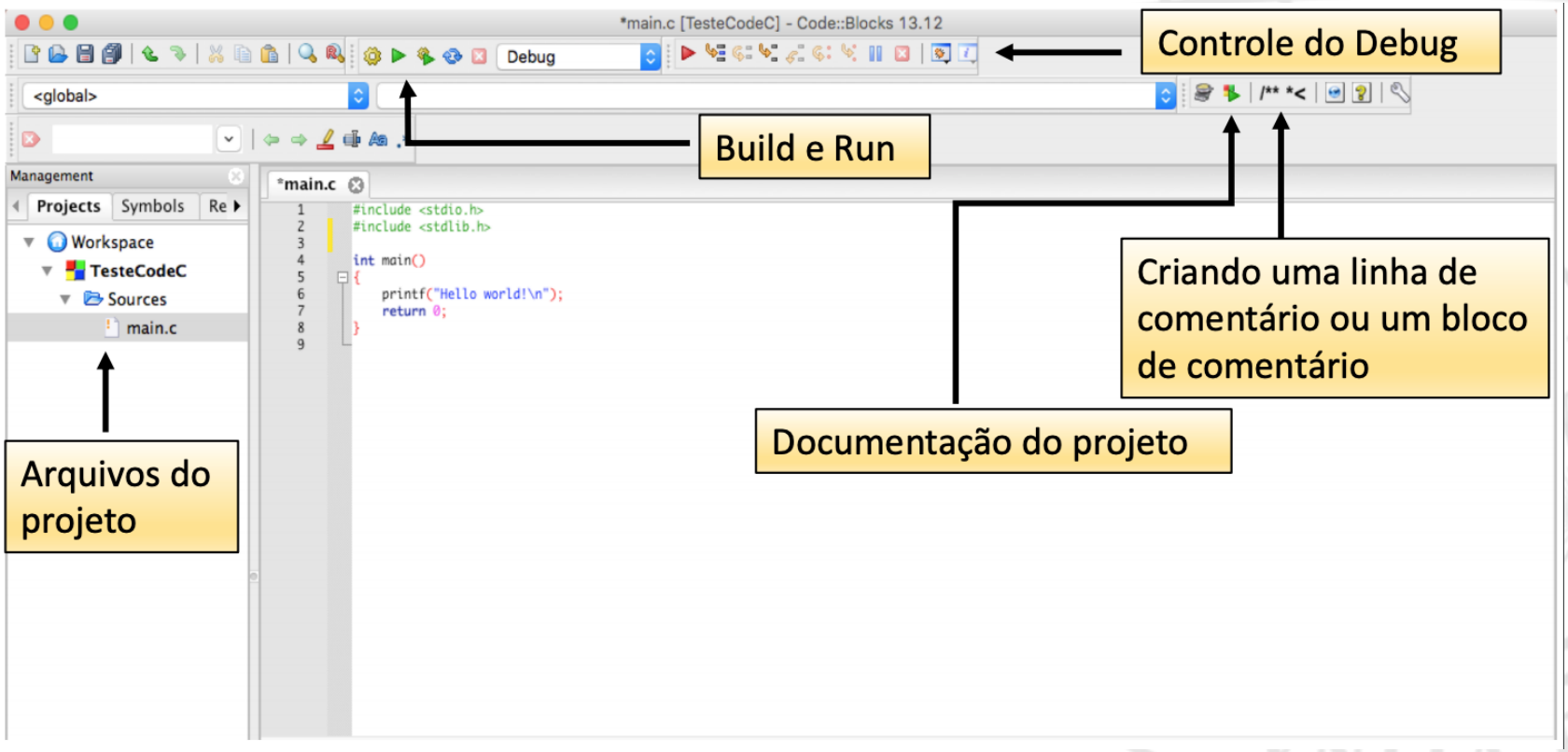

DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto

- Por fim, pode-se utilizar as seguintes opções do menu **Build** para compilar e executar o programa
  - **Compile current file (Ctrl + Shift + F9)**
    - Opção que transforma seu arquivo de código-fonte em instruções de máquina e gera um arquivo do tipo objeto
  - **Build (Ctrl + F9)**
    - Será compilado todos os arquivos do seu projeto para fazer o processo de ligação com tudo o que é necessário para gerar o executável do seu programa
  - **Build and run (F9)**
    - Além de gerar o executável, essa opção também executa o programa gerado

DEPARTAMENTO DE COMPUTAÇÃO

# Funcionalidades




DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Com o passar do tempo, nosso conhecimento sobre programação cresce, assim como a complexidade de nossos programas
- Surge a necessidade de examinar o nosso programa à procura de erros ou defeitos no código-fonte
- Para realizar essa tarefa, contamos com a ajuda de um **debugger** ou **depurador**

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- O debugger nada mais é que um programa de computador usado para testar e depurar (limpar) outros programas
- Entre as principais funcionalidades de um **debugger** estão:
  - Possibilidade de executar o programa passo a passo
  - Pausar o programa em pontos predefinidos, chamados pontos de paradas (**breakpoints**), para examinar o estado atual de suas variáveis
- Todas as funcionalidades do **debugger** podem ser encontradas no menu **Debug**

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Exemplo de uso do debugger:
- Primeiramente, vamos colocar dois pontos de parada no programa, nas linhas 13 e 23
  - Isso pode ser feito clicando no lado direito do número da linha

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Iniciamos o debugger com a opção **Start (F8)**
- Isso fará com que o programa seja executado normalmente até encontrar um **breakpoint**

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Note que existe um **triângulo amarelo** dentro do primeiro **breakpoint**
- Esse triângulo indica em que parte do programa esta parado

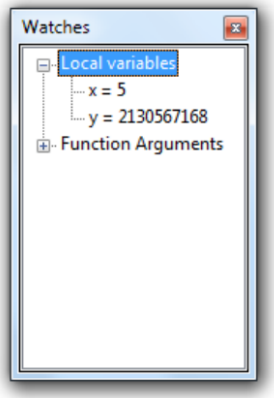
```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Dentro da opção **Debugging Windows**, podemos habilitar a opção **Watches**
  - Essa opção vai abrir uma pequena janela que permite visualizar o valor atual das variáveis de um programa, assim como os valores passados para funções

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```




DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- A partir de determinado ponto do programa, pode-se mover para a próxima linha do programa com a opção **Next line (F7)**
- O programa será executado passo a passo, sempre avançando para a linha seguinte do escopo onde esta

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Se houver uma chamada de função (linha 15) a opção **Next line (F7)** chama a função, mas não permite que a estudemos passo a passo
- Para entrar dentro do código de uma função, utilizamos a opção **Step Into (Shift+F7)** na linha da chamada da função

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

- Pode-se percorrer a função passo a passo com a opção **Next line (F7)**
  - Terminada a função, o **debugger** vai para a linha seguinte ao ponto do código chamante
  - Para ignorar o resto da função e voltar para onde original no código que chamou a função, basta clicar **Step out (Shift+Ctrl+F7)**

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Debugger

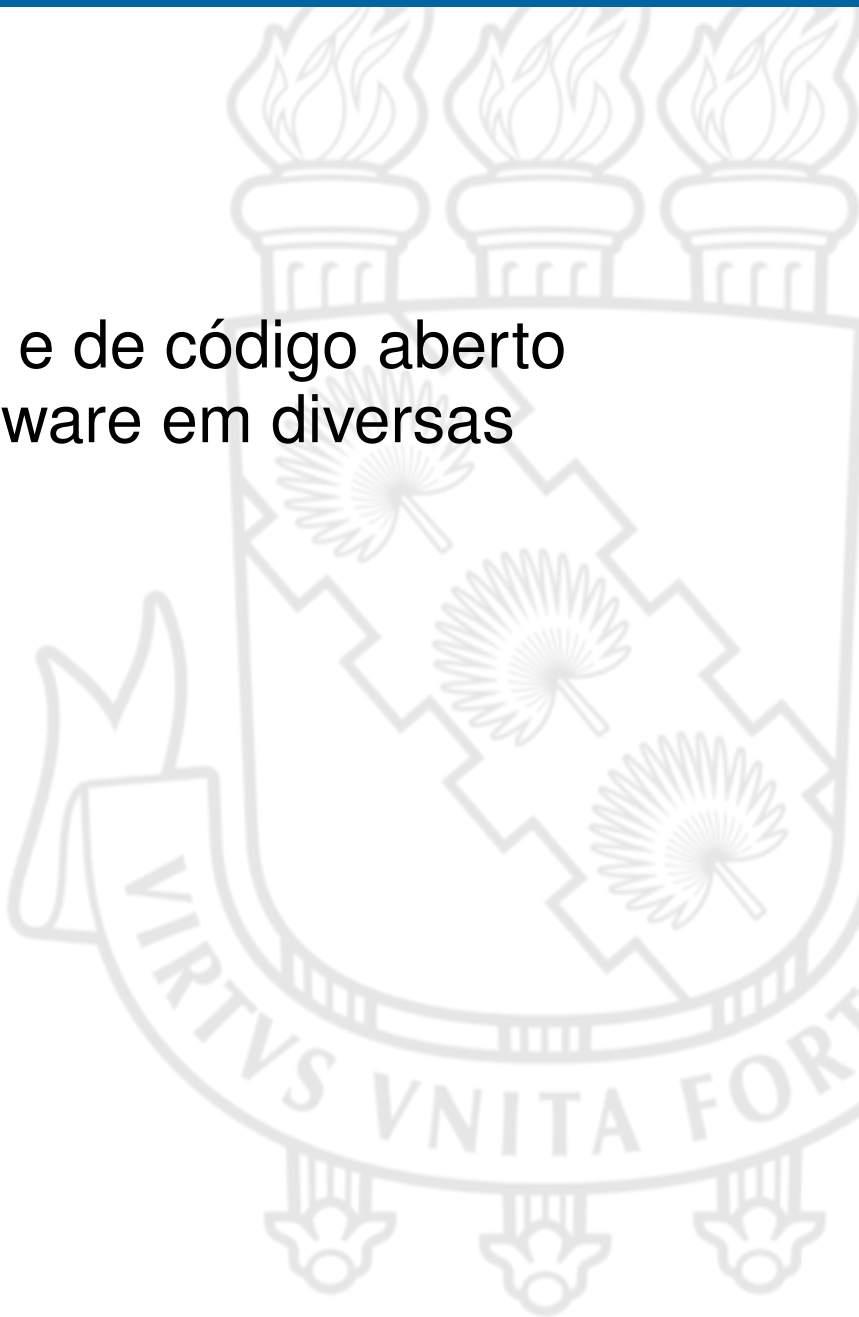
- Para avançar todo o código e ir direto para o próximo **breakpoint** (linha 23), podemos usar a opção **Continue (Ctrl+F7)**
- Por fim, para parar o **debugger**, basta clicar na opção **Stop debugger**

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int fatorial(int n){
4       int i,f = 1;
5       for (i = 1; i <= n; i++)
6           f = f * i;
7       return f;
8   }
9   int main(){
10      int x,y;
11      printf("Digite um valor inteiro: ");
12      scanf("%d",&x);
13      if (x > 0){
14          printf("X eh positivo\n");
15          y = fatorial(x);
16          printf("Fatorial de X eh %d\n",y);
17      }else{
18          if (x < 0)
19              printf("X eh negativo\n");
20          else
21              printf("X eh Zero\n");
22      }
23      printf("Fim do programa!\n");
24      system("pause");
25      return 0;
26  }
27
```

DEPARTAMENTO DE COMPUTAÇÃO

# Netbeans

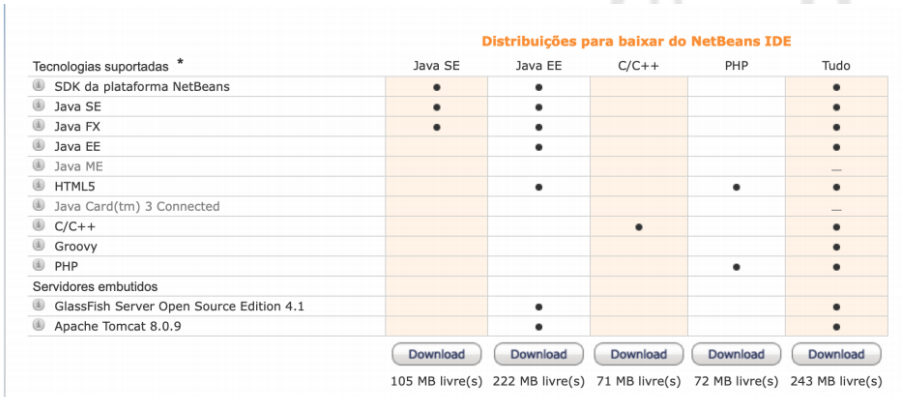
- NetBeans é uma IDE gratuita e de código aberto para desenvolvimento de software em diversas linguagens:
  - Java
  - HTML5
  - C/C++
  - PHP
  - ...




DEPARTAMENTO DE COMPUTAÇÃO

# Netbeans

- O site do netbeans: `https://netbeans.org`
- O download pode ser feito pelo link:
  - `https://netbeans.org/downloads/8.0.1/?pagelang=pt_BR`
  - Existe uma versão própria para desenvolvimento de aplicações C/C++
  - Porém, iremos usar a versão completa

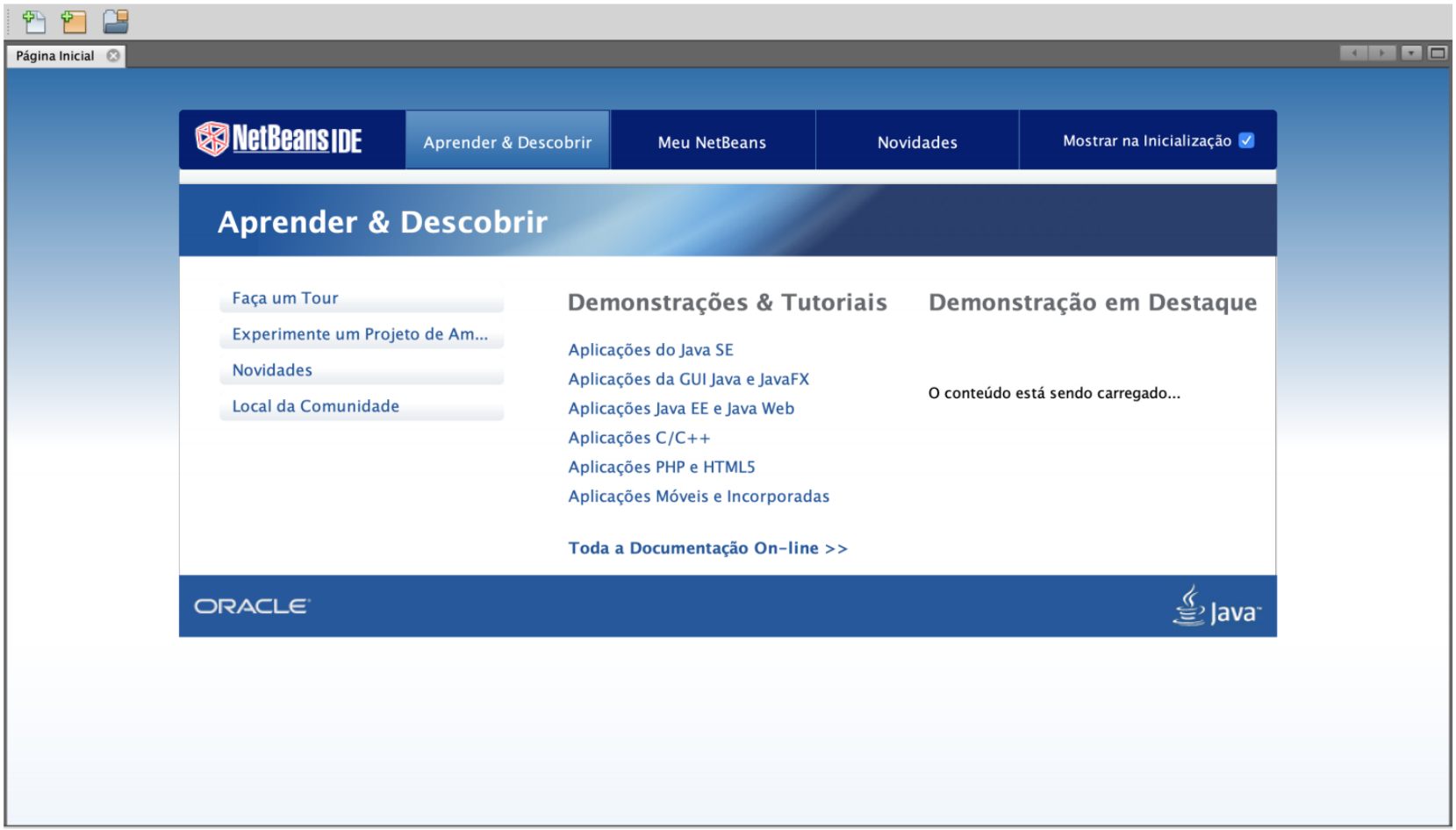


Distribuições para baixar do NetBeans IDE

| Tecnologias suportadas * | Java SE | Java EE | C/C++ | PHP | Tudo |
|---|---|---|---|---|---|
| SDK da plataforma NetBeans | • | • | | | • |
| Java SE | • | • | | | • |
| Java FX | • | • | | | • |
| Java EE | | • | | | • |
| Java ME | | | | | – |
| HTML5 | | • | | • | • |
| Java Card(tm) 3 Connected | | | | | – |
| C/C++ | | | • | | • |
| Groovy | | | | | • |
| PHP | | | | • | • |
| Servidores embutidos | | | | | |
| GlassFish Server Open Source Edition 4.1 | | • | | | • |
| Apache Tomcat 8.0.9 | | • | | | • |
| | Download | Download | Download | Download | Download |
| | 105 MB livre(s) | 222 MB livre(s) | 71 MB livre(s) | 72 MB livre(s) | 243 MB livre(s) |

DEPARTAMENTO DE COMPUTAÇÃO

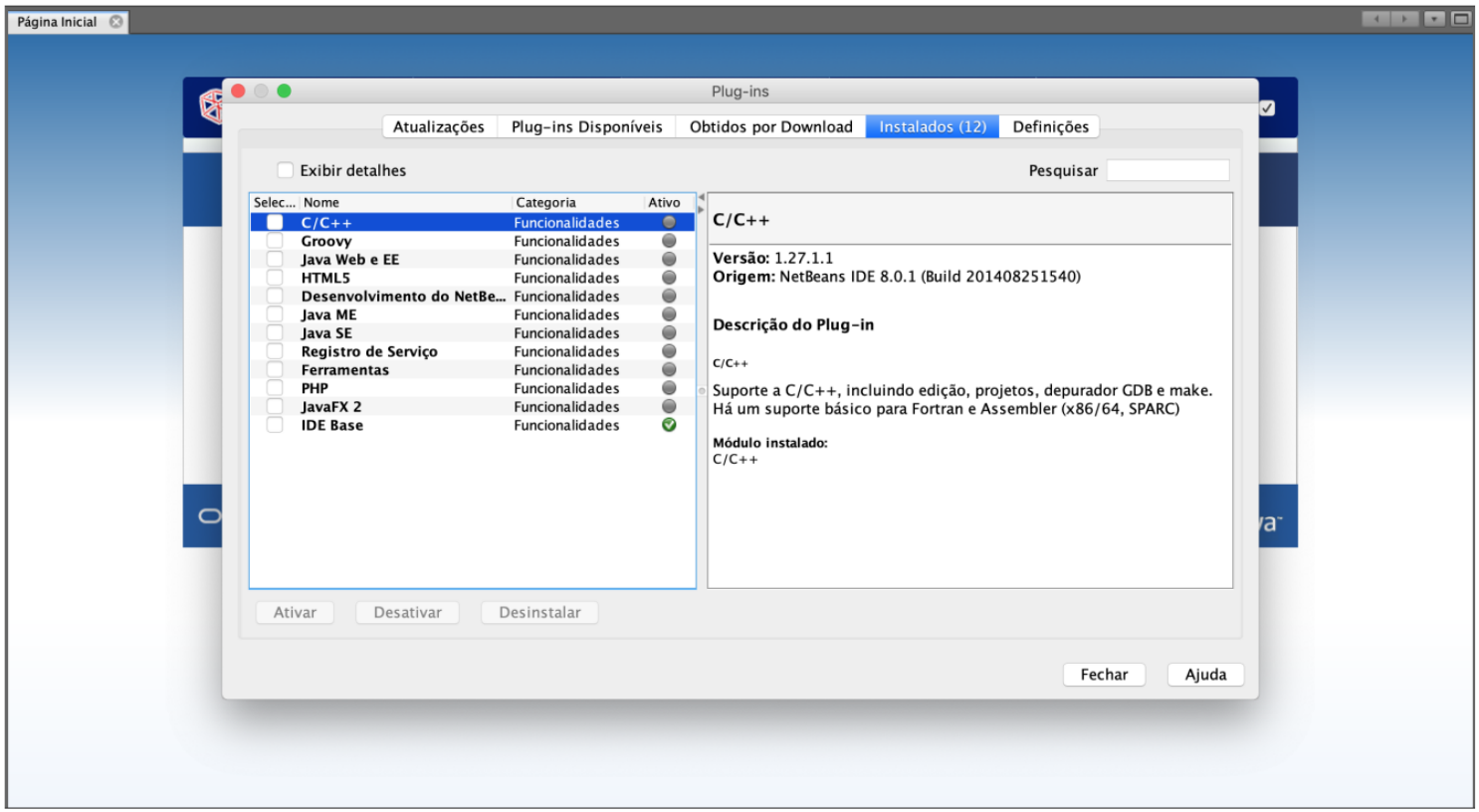
# Configurando o Netbeans




DEPARTAMENTO DE COMPUTAÇÃO

# Configurando o Netbeans

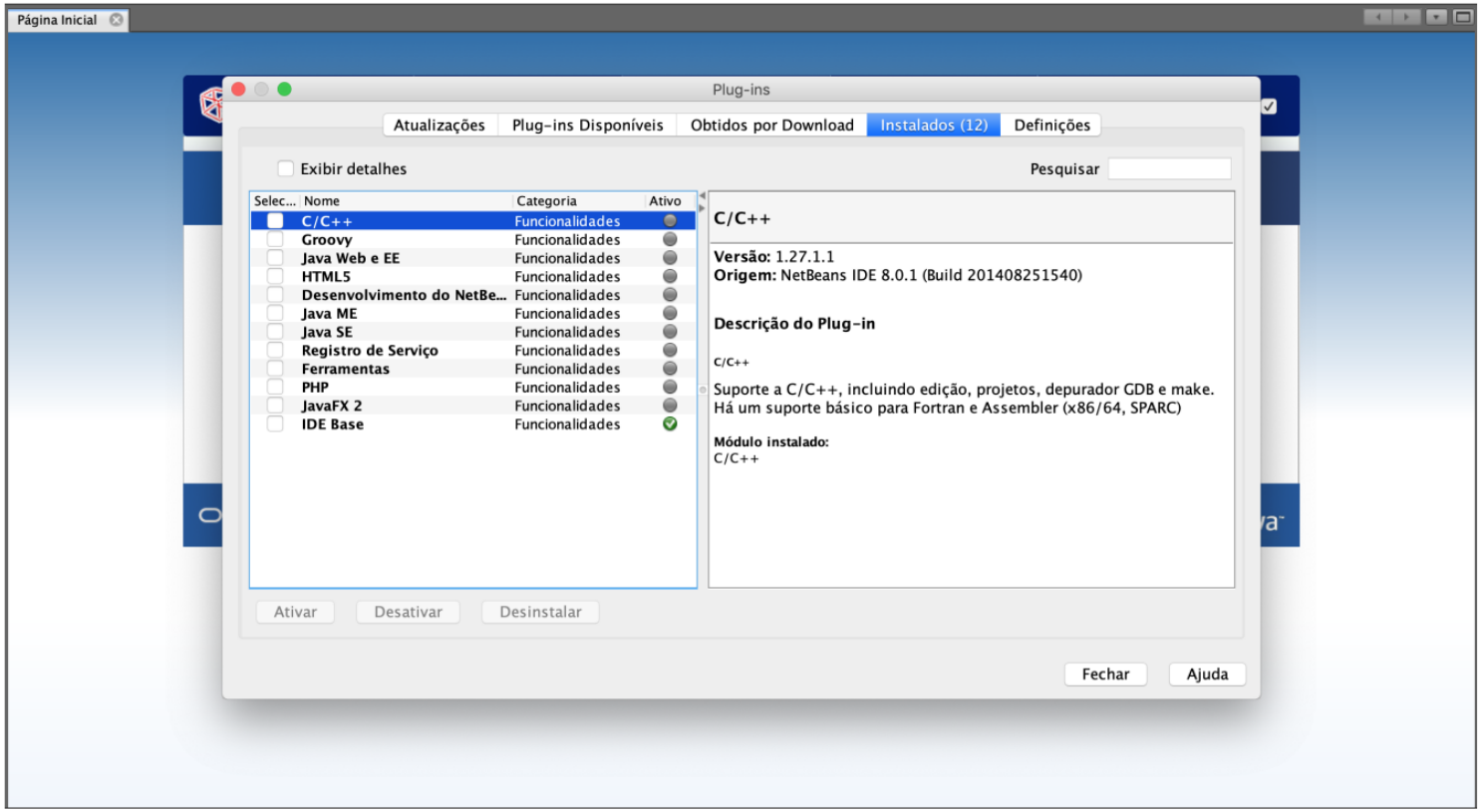
- Instalar **Plug-ins** → instalados




DEPARTAMENTO DE COMPUTAÇÃO

# Configurando o Netbeans

- Escolha: **C/C++** → Ativar

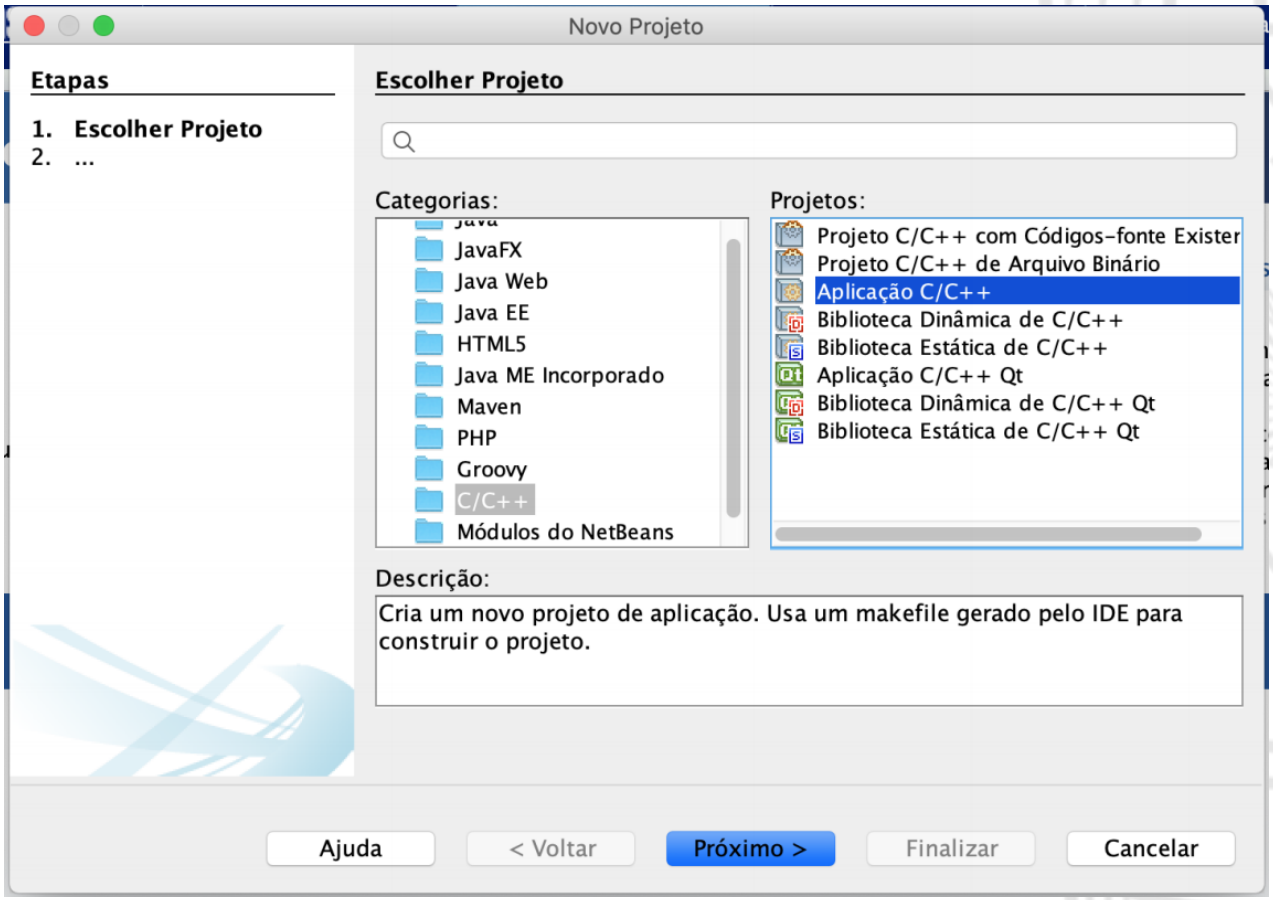



DEPARTAMENTO DE COMPUTAÇÃO

# Configurando o Netbeans

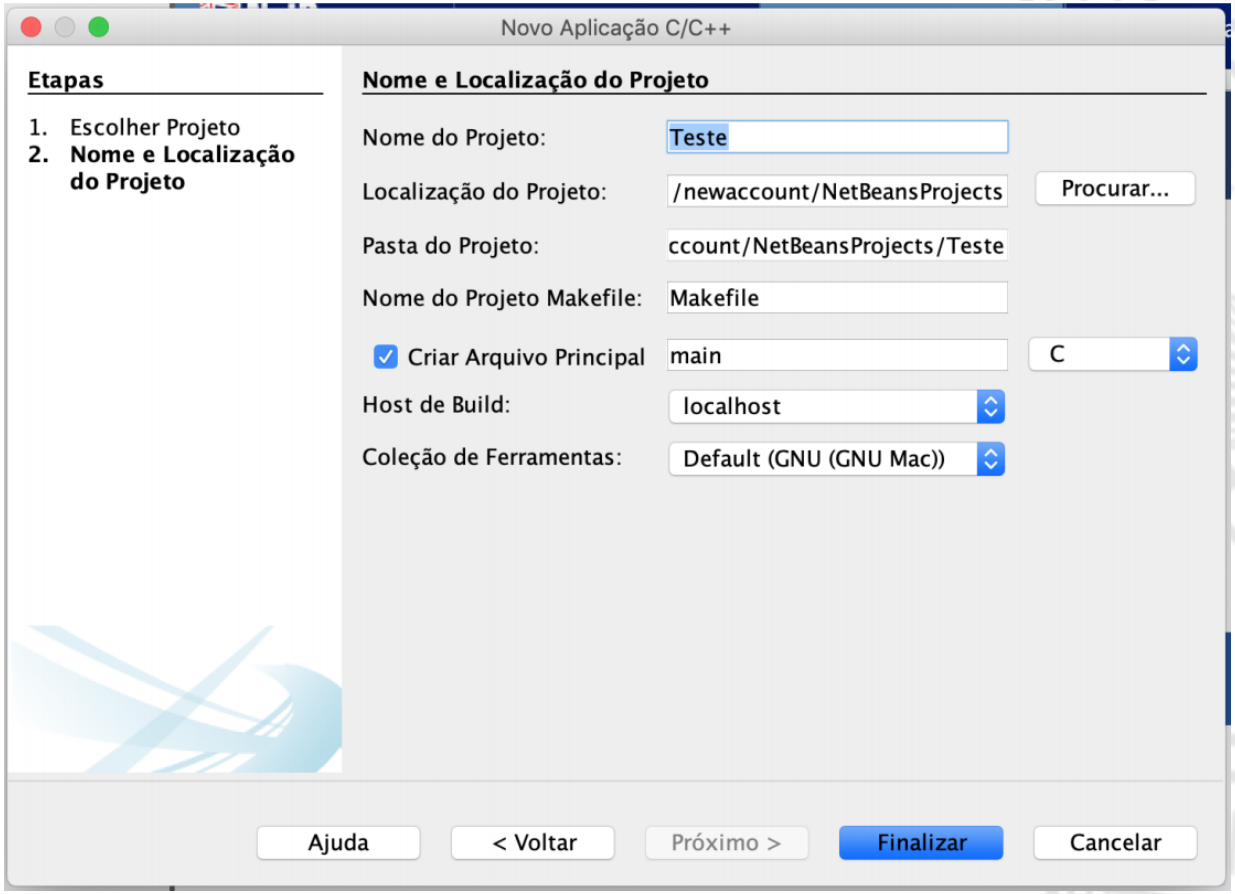
- Espere a instalação do **Plug-in** e pronto!
- Agora o NetBeans já está pronto para o desenvolvimento de aplicações C/C++
- Reinicie o NetBeans e crie um projeto

# Criando um Projeto



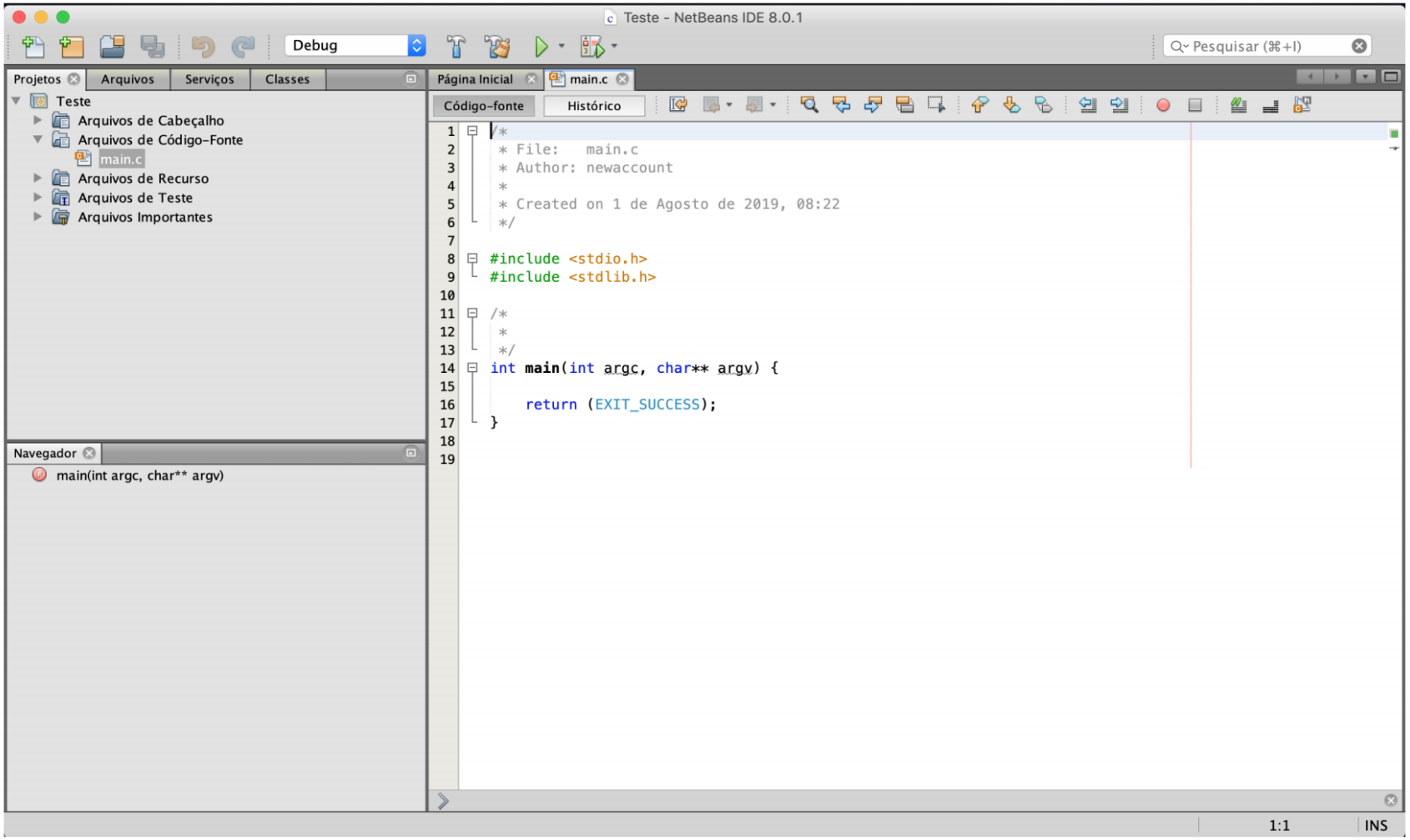

DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto




DEPARTAMENTO DE COMPUTAÇÃO

# Criando um Projeto



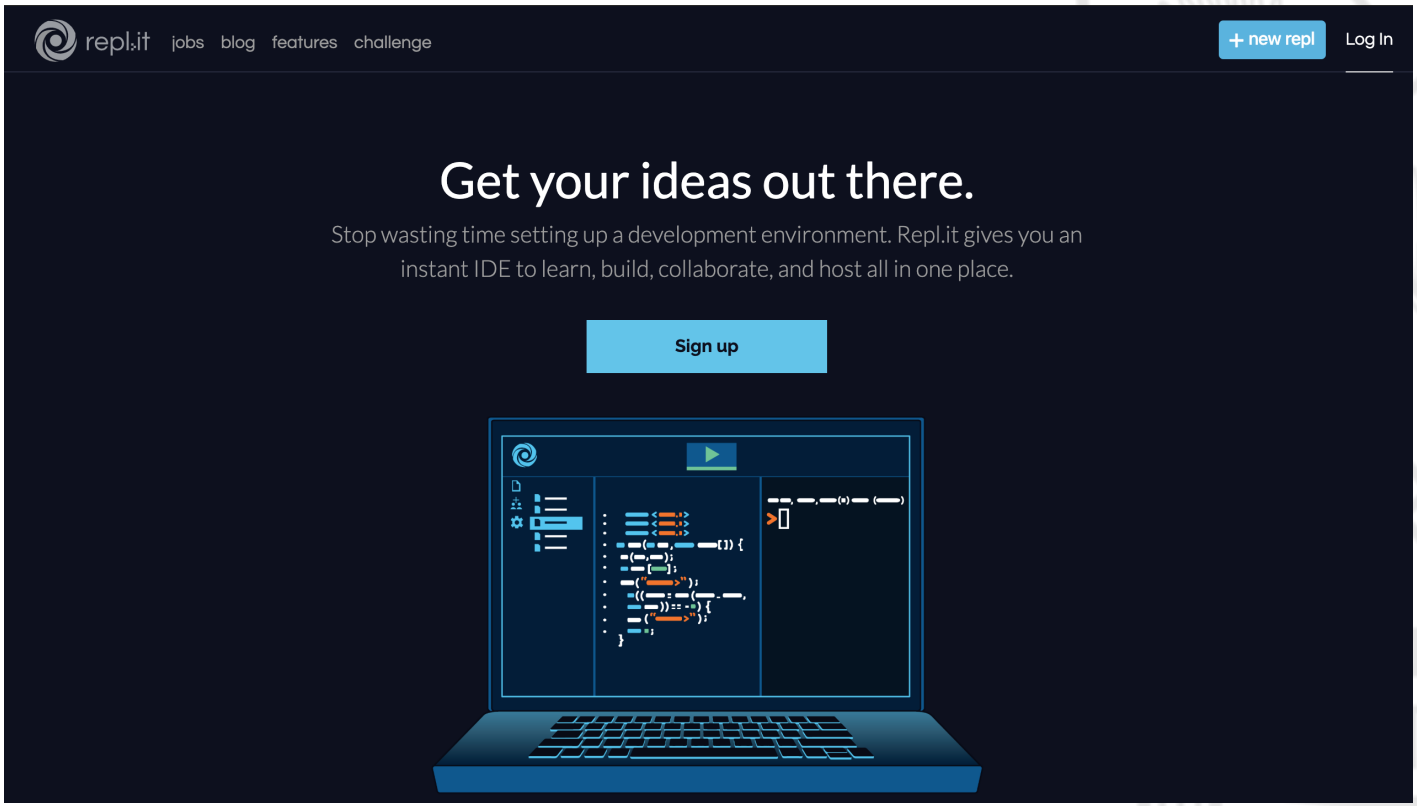
# Repl.it

- É uma IDE online que suporta diversas linguagens de programação
- O ambiente de desenvolvimento pronto para a criação de programas
- É possível usar a ferramenta sem ter um cadastro
- Site da Repl.it: `https://repl.it/`

DEPARTAMENTO DE COMPUTAÇÃO

# Repl.it

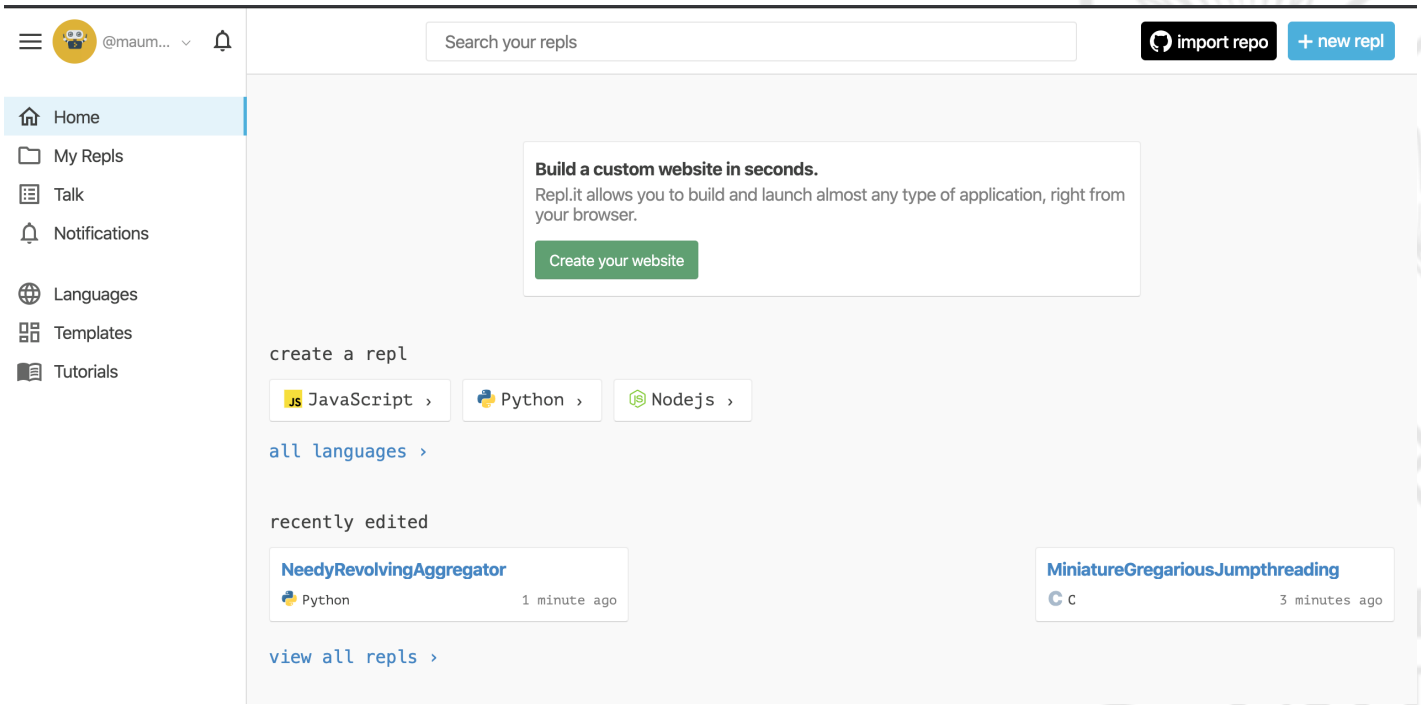
- Clique em **Sign up**




DEPARTAMENTO DE COMPUTAÇÃO

# Repl.it

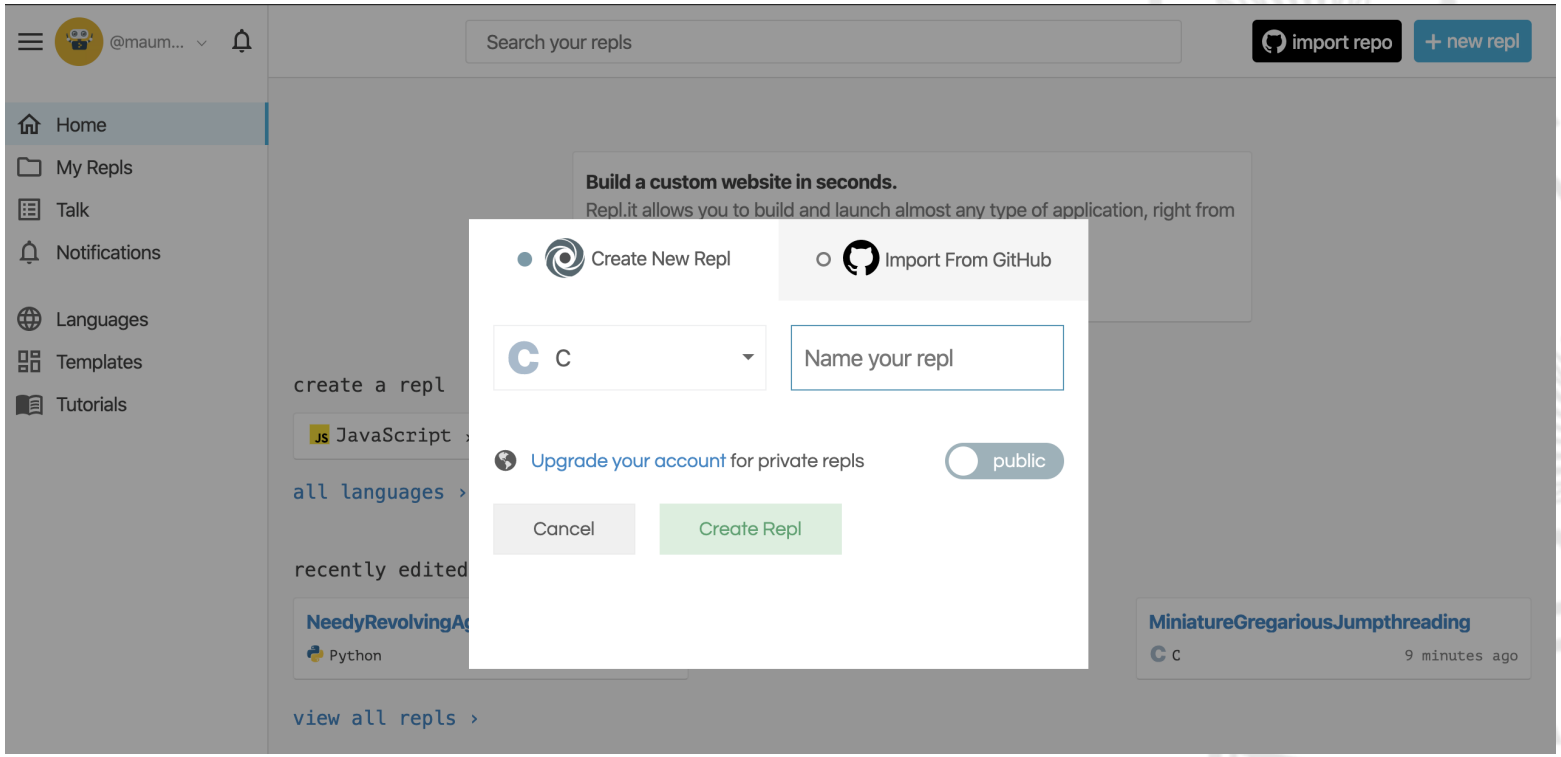
- Clique em **+ new repl**




DEPARTAMENTO DE COMPUTAÇÃO

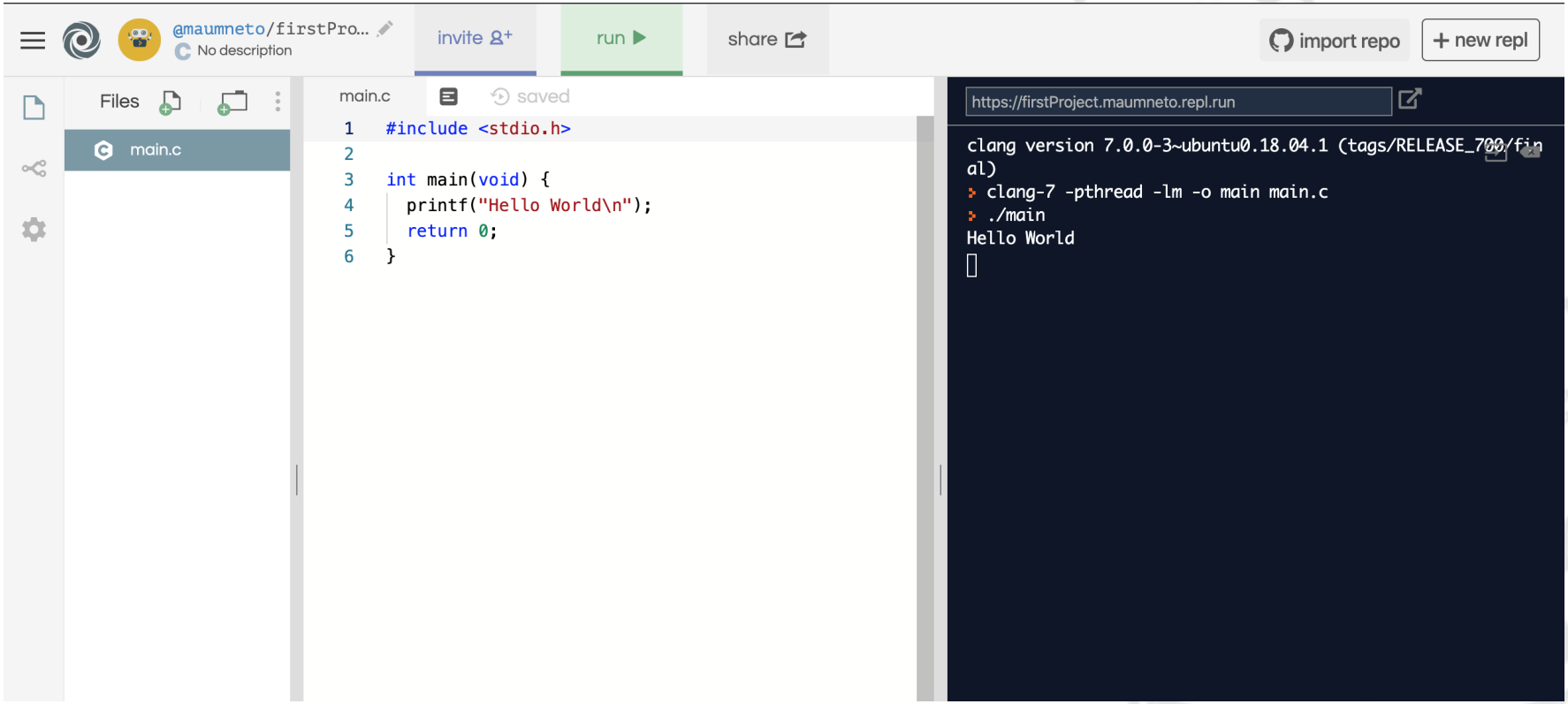
# Repl.it

- **C language** → nomeie o projeto → **Create Repl**




DEPARTAMENTO DE COMPUTAÇÃO

# Repl.it




DEPARTAMENTO DE COMPUTAÇÃO