

PROGRAMAÇÃO COMPUTACIONAL PARA ENGENHARIA

INTEGRATED DEVELOPMENT ENVIRONMENT

Maurício Moreira Neto¹

¹Universidade Federal do Ceará
Departamento de Computação

30 de janeiro de 2020

Sumário

- 1 Objetivos
- 2 Definição
- 3 Code Blocks

- 4 Debugger
- 5 Netbeans
- 6 Configurando Ambiente
- 7 Repl.it



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

Definição

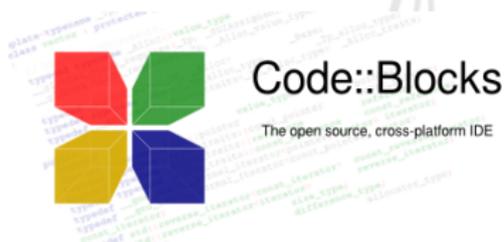
- IDE significa Ambiente de Desenvolvimento Integrado
 - Do inglês – Integrated Development Environment
- As IDEs são utilizadas para criar programas em diversas linguagens de programação, contendo ferramentas necessárias para o desenvolvimento do software

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**

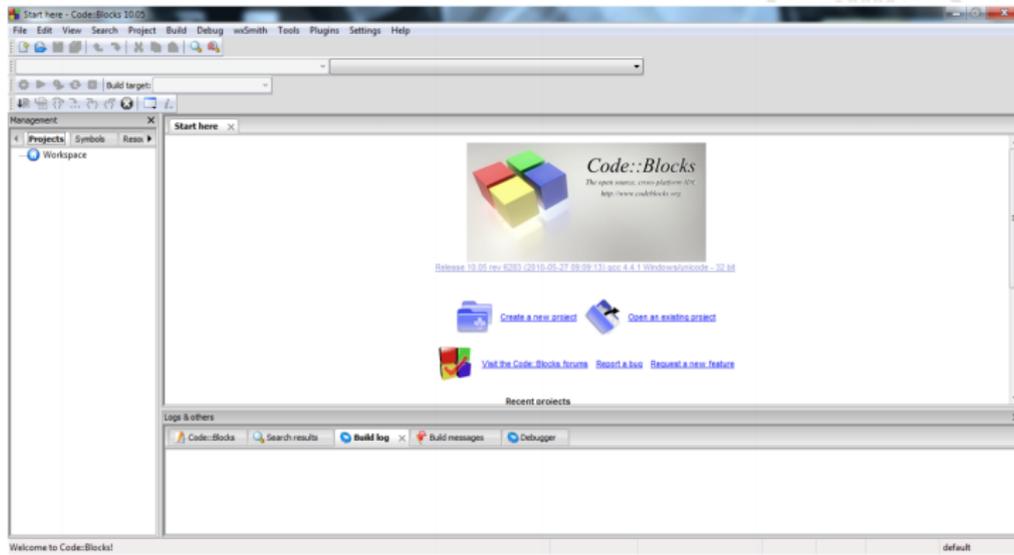
Code Blocks

- 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**



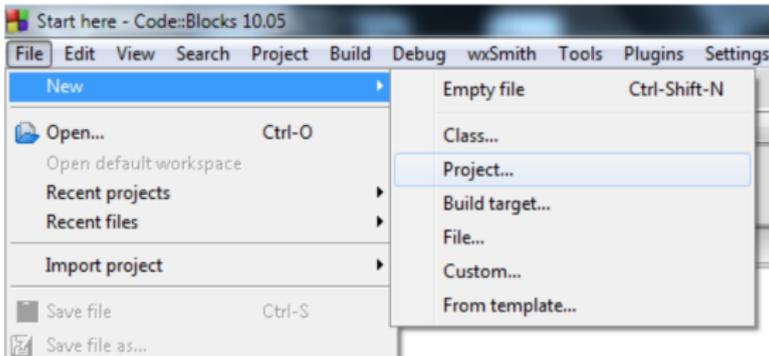
Criando um Projeto

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



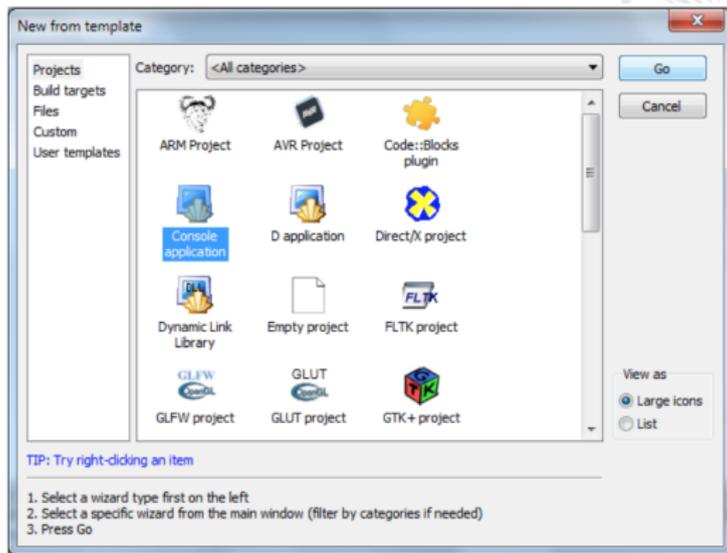
Criando um Projeto

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



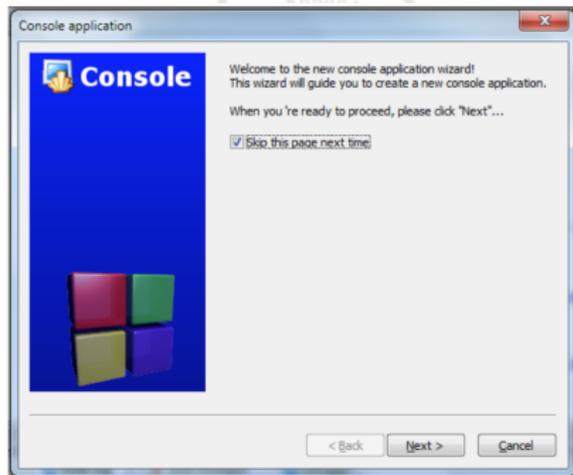
Criando um Projeto

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



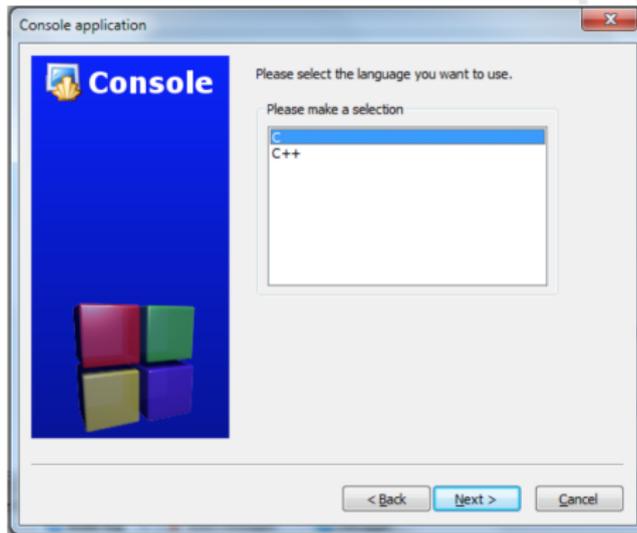
Criando um Projeto

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



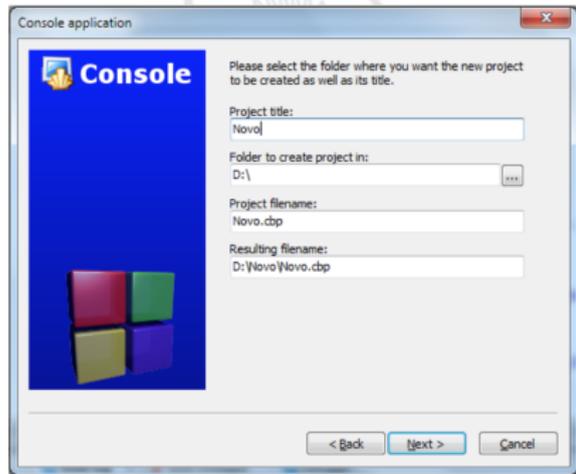
Criando um Projeto

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



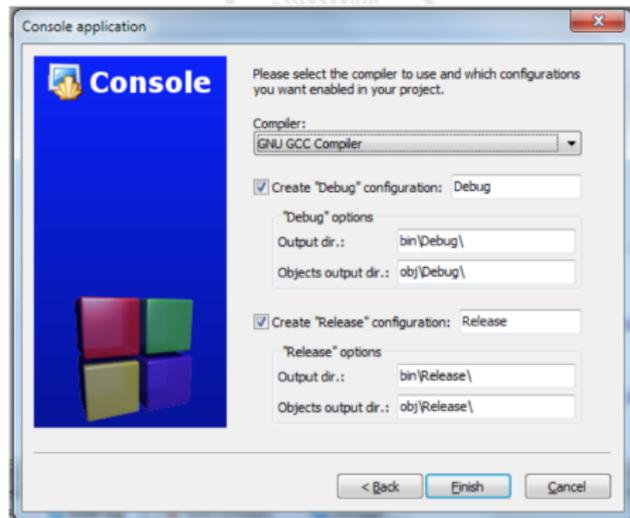
Criando um Projeto

- **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**



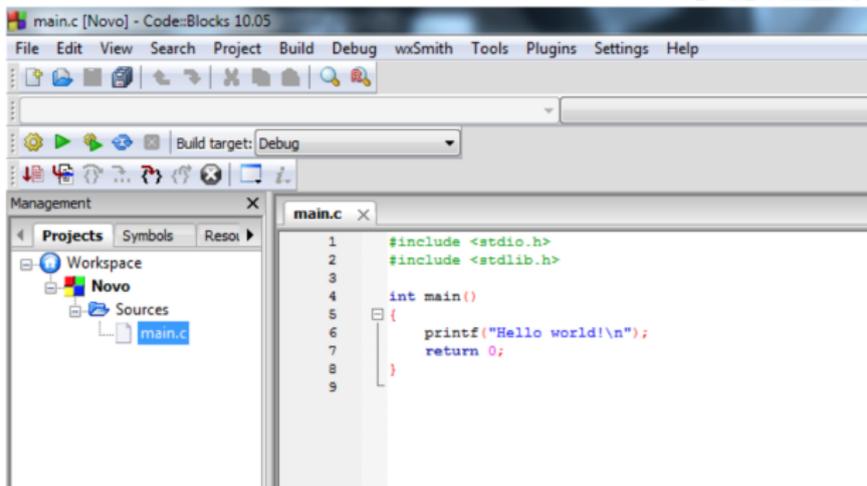
Criando um Projeto

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



Criando um Projeto

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



The screenshot shows the Code::Blocks IDE interface. The main window displays a C program in a file named 'main.c'. The code is as follows:

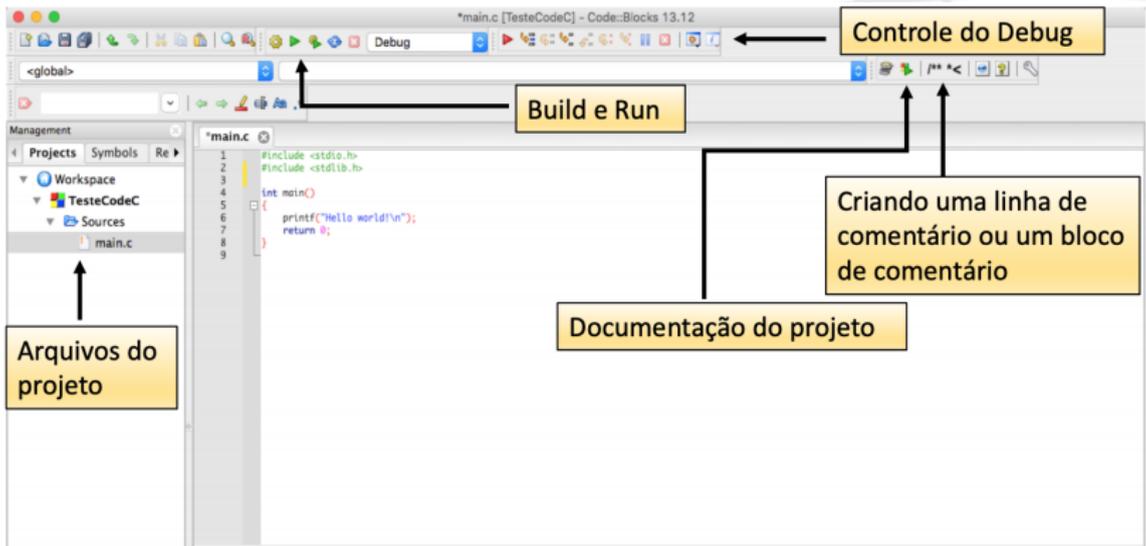
```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     printf("Hello world!\n");
7     return 0;
8 }
9
```

The left sidebar shows the 'Management' view with a 'Projects' pane. A project named 'main.c' is visible under a 'Sources' folder. The top menu bar includes 'File', 'Edit', 'View', 'Search', 'Project', 'Build', 'Debug', 'wxSmith', 'Tools', 'Plugins', 'Settings', and 'Help'. The 'Build target' is set to 'Debug'.

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

Funcionalidades



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**

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**

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          }
8          return f;
9      }
10     int main(){
11         int x, y;
12         printf("Digite um valor inteiro: ");
13         scanf("%d", &x);
14         if (x > 0){
15             printf("X eh positivo\n");
16             y = fatorial(x);
17             printf("Fatorial de X eh %d\n", y);
18         }else{
19             if (x < 0)
20                 printf("X eh negativo\n");
21             else
22                 printf("X eh Zero\n");
23         }
24         printf("Fim do programa!\n");
25         system("pause");
26         return 0;
27     }
```

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
```

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
```

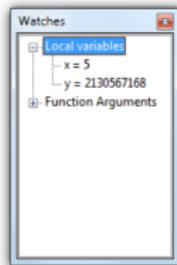
Debugger

- 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

```



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
```

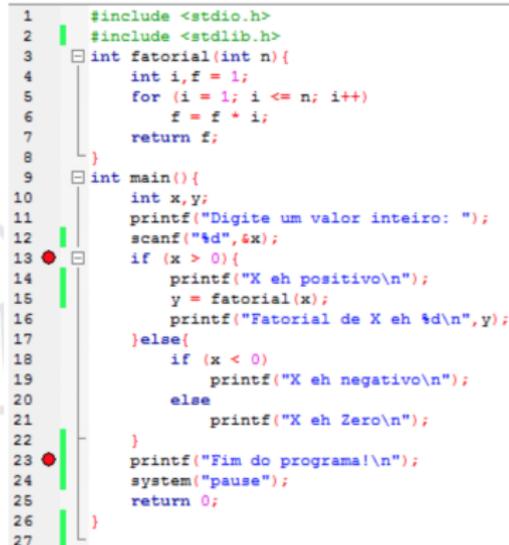
Debugger

- 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
```

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
```

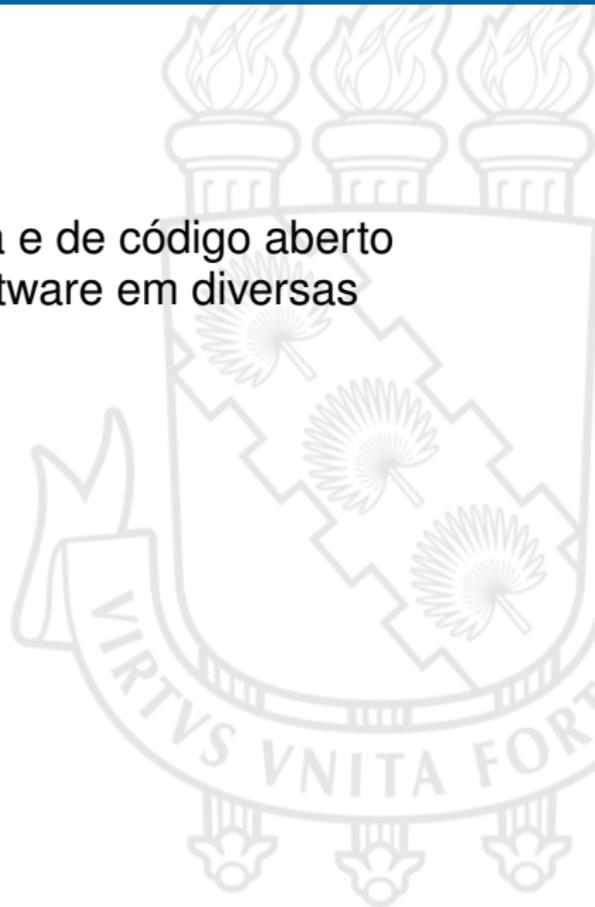
Debugger

- 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
```

Netbeans

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



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

Tecnologias suportadas *

Distribuições para baixar do NetBeans IDE

	Java SE	Java EE	C/C++	PHP	Tudo
④ SDK da plataforma NetBeans	●	●			●
④ Java SE	●	●			●
④ Java FX	●				●
④ Java EE		●			●
④ Java ME					●
④ HTML5					●
④ Java Card(tm) 3 Connected		●			●
④ C/C++			●		●
④ Groovy					●
④ PHP				●	●
Serviços embutidos					●
④ GlassFish Server Open Source Edition 4.1		●			●
④ 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)

Configurando o Netbeans



Página Inicial

NetBeans IDE Aprender & Descobrir Meu NetBeans Novidades Mostrar na Inicialização

Aprender & Descobrir

Faça um Tour

Experimente um Projeto de Am...

Novidades

Local da Comunidade

Demonstrações & Tutoriais

Aplicações do Java SE

Aplicações da GUI Java e JavaFX

Aplicações Java EE e Java Web

Aplicações C/C++

Aplicações PHP e HTML5

Aplicações Móveis e Incorporadas

Toda a Documentação On-line >>

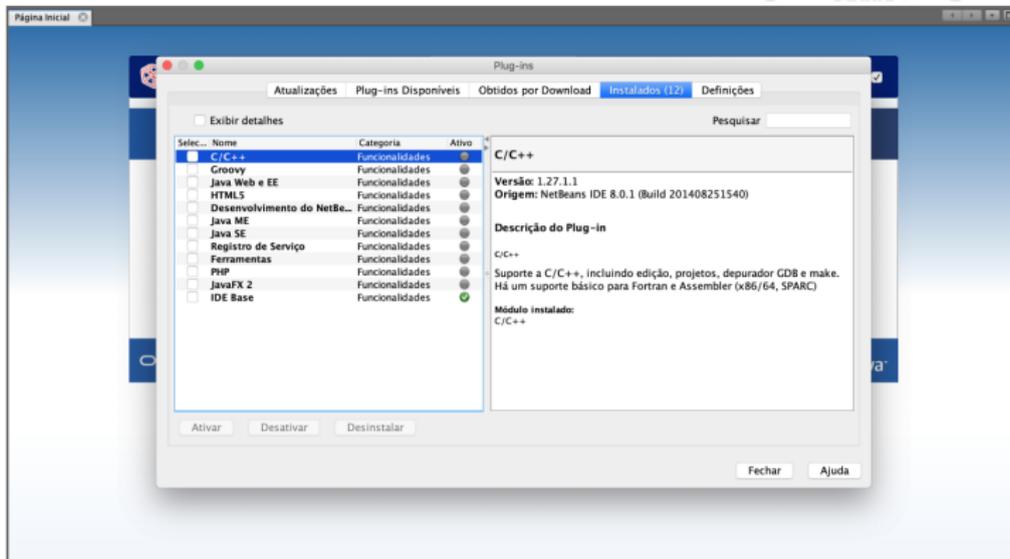
Demonstração em Destaque

O conteúdo está sendo carregado...

ORACLE 

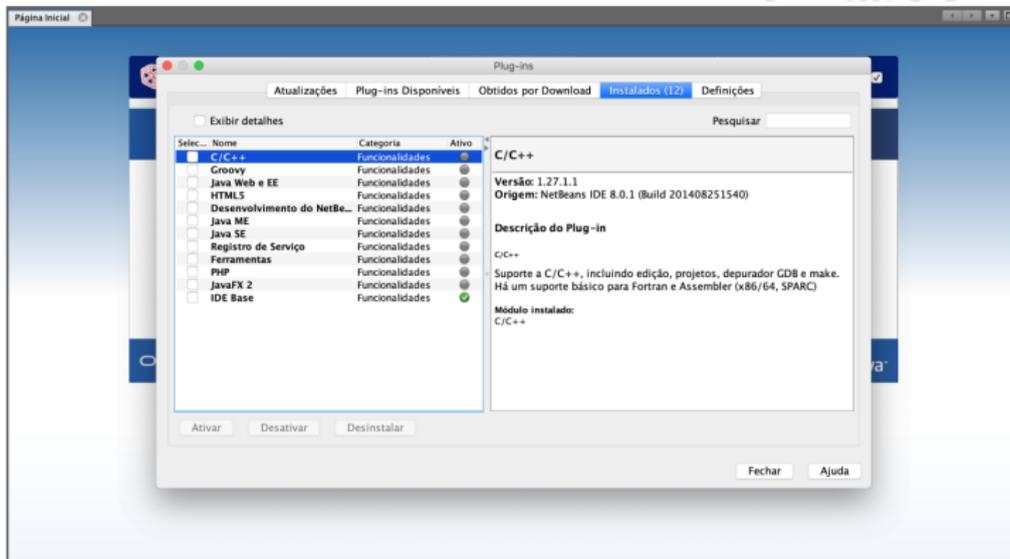
Configurando o Netbeans

- Instalar **Plug-ins** → instalados



Configurando o Netbeans

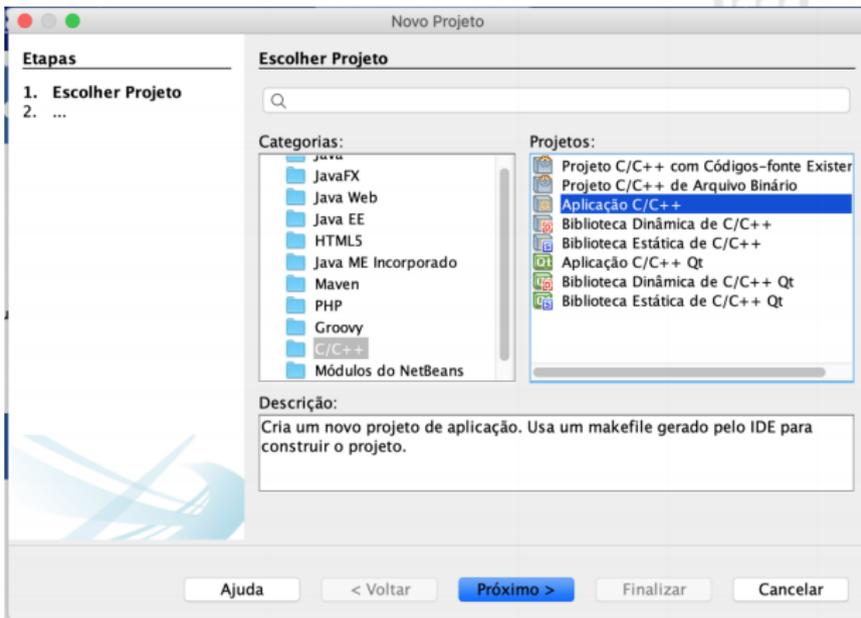
- Escolha: **C/C++** → Ativar



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



Criando um Projeto

Novo Aplicação C/C++

Etapas

1. Escolher Projeto
2. **Nome e Localização do Projeto**

Nome e Localização do Projeto

Nome do Projeto:

Localização do Projeto: Procurar...

Pasta do Projeto:

Nome do Projeto Makefile:

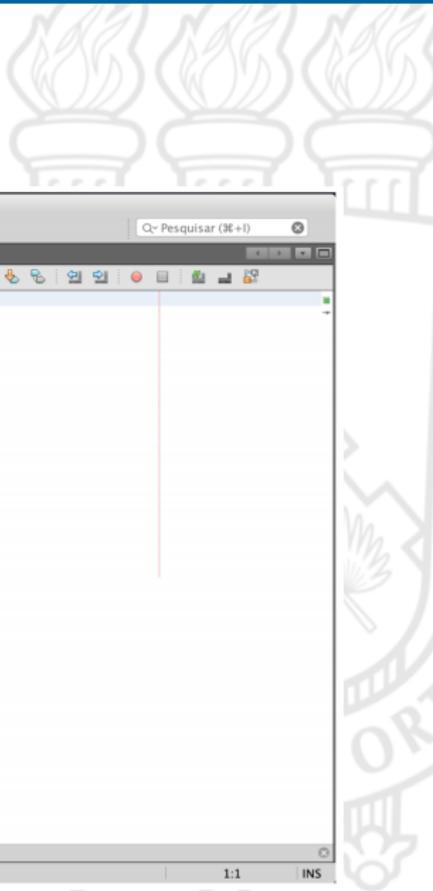
Criar Arquivo Principal

Host de Build:

Coleção de Ferramentas:

Ajuda < Voltar Próximo > Finalizar Cancelar

Criando um Projeto



```
1  /*  
2  * File:   main.c  
3  * Author: newaccount  
4  *  
5  * Created on 1 de Agosto de 2019, 08:22  
6  */  
7  
8  #include <stdio.h>  
9  #include <stdlib.h>  
10  
11 /*  
12 */  
13 */  
14 int main(int argc, char** argv) {  
15     return (EXIT_SUCCESS);  
16 }  
17  
18  
19
```

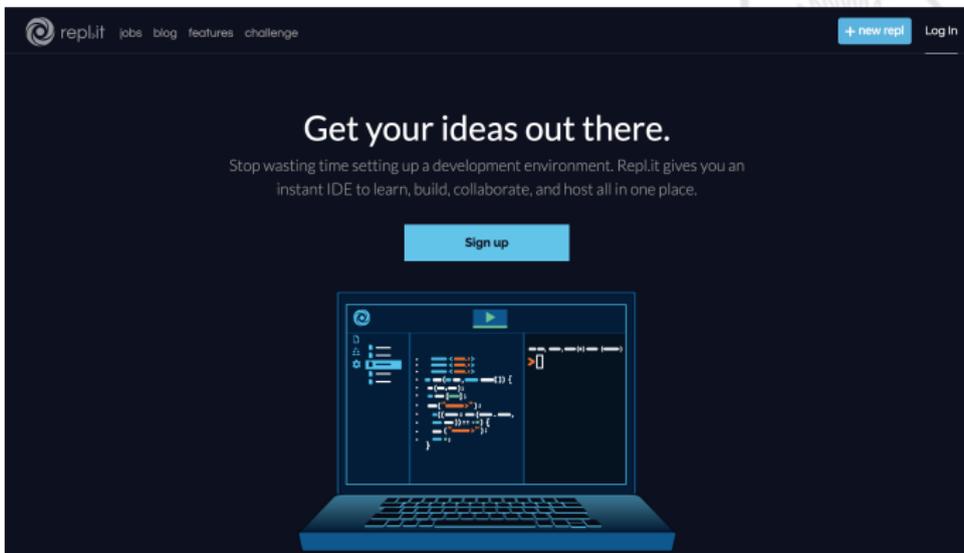


Repl.it

- 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/>

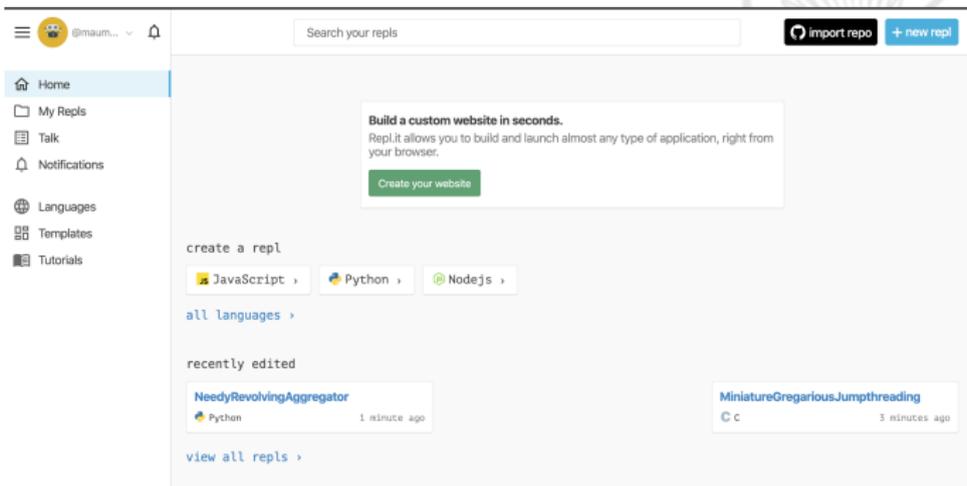
Repl.it

■ Clique em **Sign up**



Repl.it

■ Clique em + new repl



The screenshot shows the Repl.it homepage. At the top right, there is a search bar and two buttons: "import repo" and "+ new repl". The left sidebar contains navigation links: Home, My Repls, Talk, Notifications, Languages, Templates, and Tutorials. The main content area features a central card with the text "Build a custom website in seconds. Repl.it allows you to build and launch almost any type of application, right from your browser." and a "Create your website" button. Below this, there is a "create a repl" section with buttons for JavaScript, Python, and Node.js. Further down, a "recently edited" section displays two repls: "NeedyRevolvingAggregator" (Python, 1 minute ago) and "MiniatureGregariousJumpthreading" (C, 3 minutes ago). A "view all repls" link is at the bottom.

Repl.it

- C language → nomeie o projeto → Create Repl

Build a custom website in seconds.
Repl.it allows you to build and launch almost any type of application, right from

Create New Repl Import From GitHub

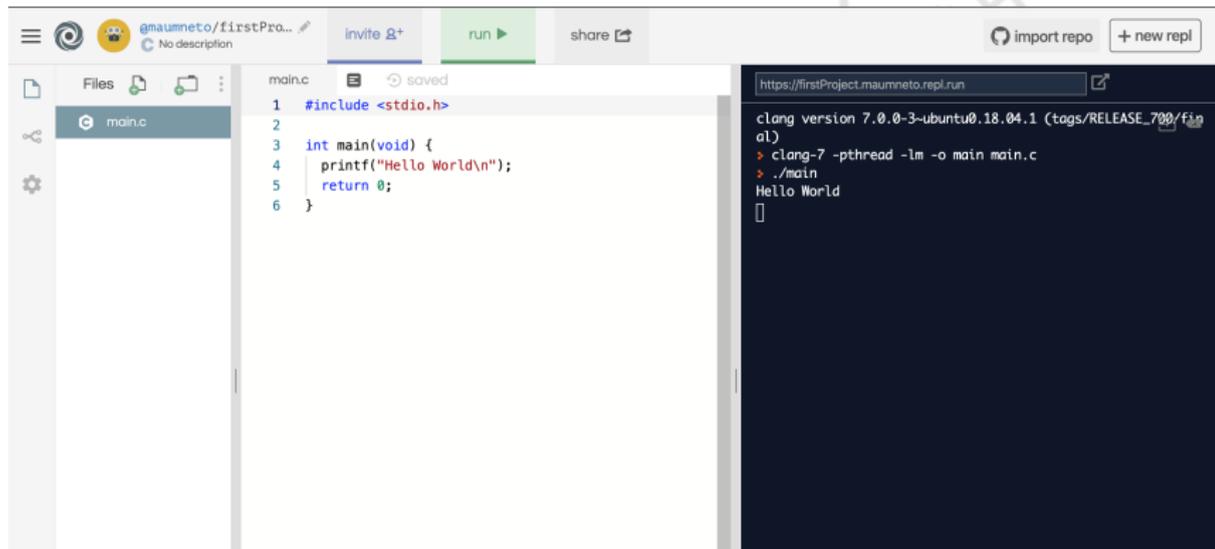
C C

[Upgrade your account for private repls](#) public

Cancel Create Repl

MiniatureGregariousJumpthreading
C C 9 minutes ago

Repl.it



The screenshot displays the Repl.it web-based IDE interface. At the top, there are navigation buttons: 'invite', 'run', and 'share'. The main workspace is divided into three sections:

- Files:** A sidebar on the left showing a file named 'main.c'.
- Code Editor:** The central area containing the following C code:

```
1 #include <stdio.h>
2
3 int main(void) {
4     printf("Hello World\n");
5     return 0;
6 }
```
- Terminal:** A dark panel on the right showing the output of the compilation and execution:

```
clang version 7.0.0-3-ubuntu0.18.04.1 (tags/RELEASE_700/final)
> clang-7 -pthread -lm -o main main.c
> ./main
Hello World
|
```