




# UTILIZANDO O CODE BLOCKS

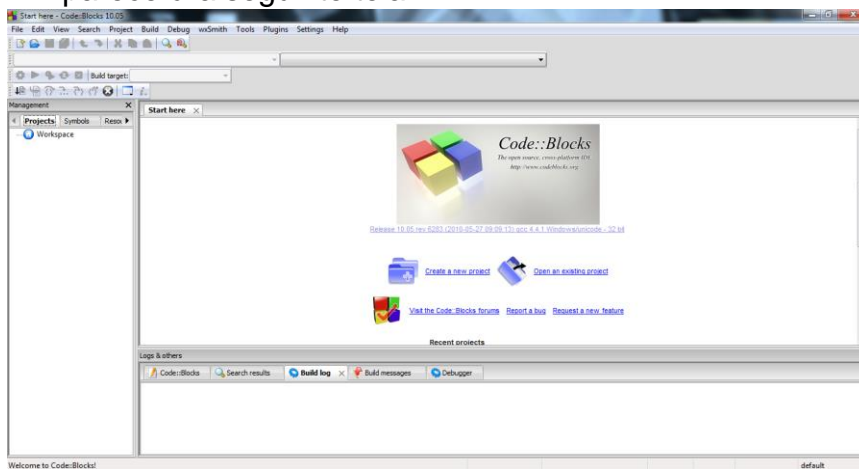
Prof. André Backes

## INTRODUÇÃO

- Existem diversos ambientes de desenvolvimento integrado ou IDEs (Integrated Development Environment) que podem ser utilizados para a programação em linguagem C.
  - Um deles é o Code::Blocks, uma IDE de código aberto e multiplataforma que suporta múltiplos compiladores.
    - O Code::Blocks pode ser baixado diretamente de seu site [www.codeblocks.org](http://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
- 

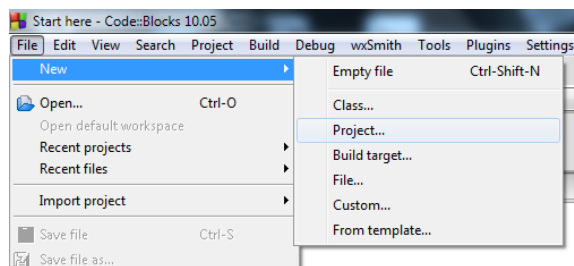
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- Primeiramente, inicie o software Code::Blocks. Aparecerá a seguinte tela



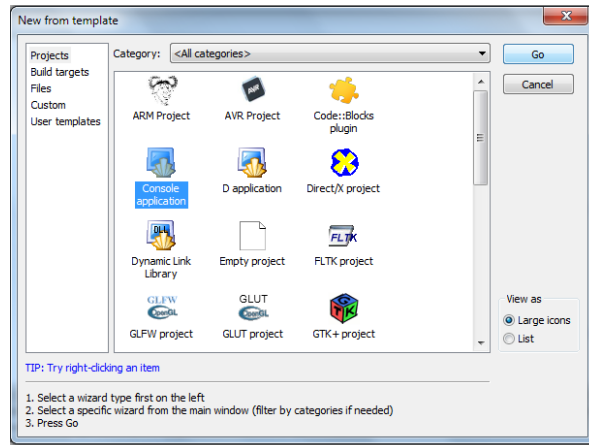
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- Em seguida clique em **File**, escolha **New** e depois **Project...**



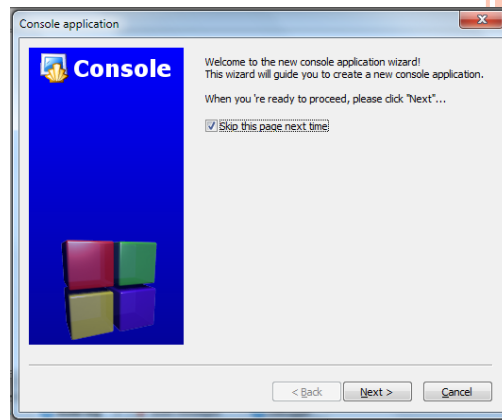
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

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



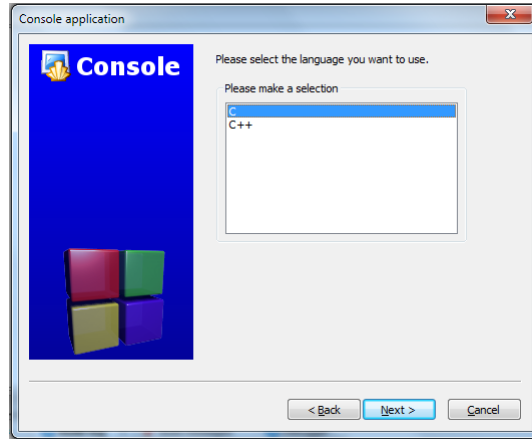
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- Caso esteja criando um projeto pela primeira vez, a tela a seguir vai aparecer.
  - Se marcarmos a opção **Skip this page next time**, essa tela de boas-vindas não será mais exibida da próxima vez que criarmos um projeto.
- Em seguida, clique em **Next**



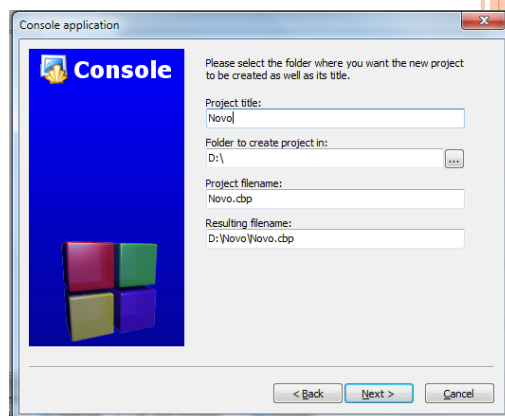
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

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



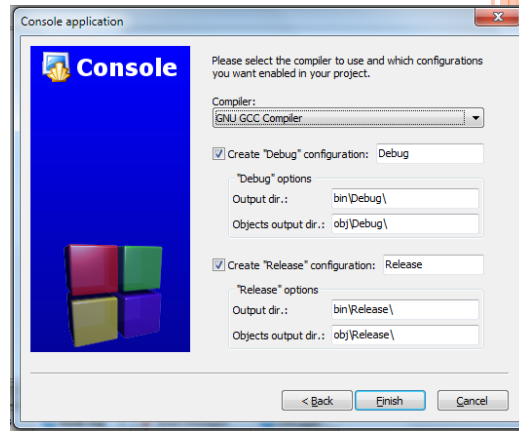
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- No campo **Project title**, coloque um nome para o seu projeto. No campo **Folder to create project in** é possível selecionar onde o projeto será salvo no computador.
  - Evite espaços e acentuação no nome e caminho do projeto
- Clique em **Next** para continuar



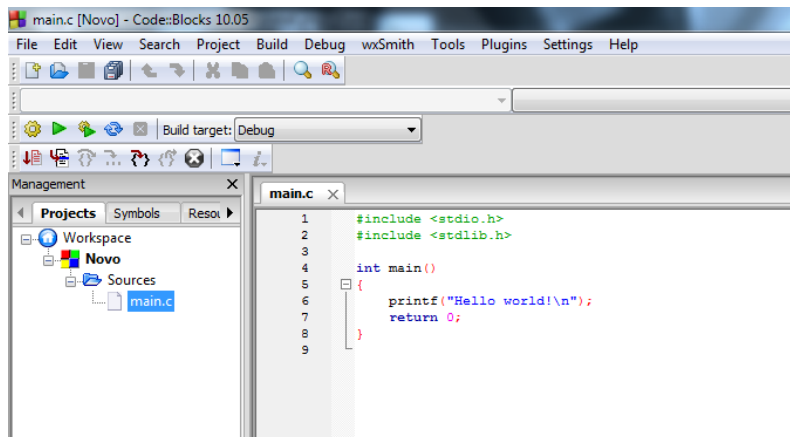
## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- Na tela a seguir, algumas configurações do compilador podem ser modificadas.
  - No entanto, isso não será necessário.
  - Basta clicar em **Finish**.



## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- Ao fim desses passos, o esqueleto de um novo programa em linguagem C terá sido criado



## CRIANDO UM NOVO PROJETO NO CODE::BLOCKS

- Por fim, podemos utilizar as seguintes opções do menu **Build** para compilar e executar nosso programa
  - **Compile current file (Ctrl+Shift+F9)**
    - essa opção vai transformar seu arquivo de código-fonte em instruções de máquina e gerar um arquivo do tipo objeto.
  - **Build (Ctrl+F9)**
    - serão compilados todos os arquivos do seu projeto para fazer o processo de “linkagem” 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.

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- Com o passar do tempo, nosso conhecimento sobre programação cresce, assim como a complexidade de nossos programas.
- Surge então 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 **depurador** ou **debugger**.

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

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



## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- Para utilizar o debugger do Code::Blocks, imagine o código ao lado
- Primeiramente, vamos colocar dois pontos de parada ou **breakpoints** 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
```

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- 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
```

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- No nosso exemplo, o usuário deverá digitar, no console, o valor lido pelo comando **scanf()** e depois retornar para a tela do **Code::Blocks** onde o programa se encontra pausado.
  - Note que existe um **triângulo amarelo** dentro do primeiro **breakpoint**.
  - Esse triângulo indica em que parte do programa a pausa está

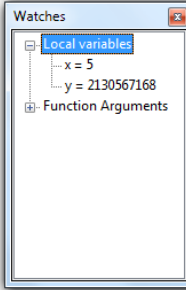
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int fatorial(int n){
4     int i, f = 1;
5     for (i = 1; i <= n; i++)
6         f = f * i;
7     return f;
8 }
9 int main(){
10    int x, y;
11    printf("Digite um valor inteiro: ");
12    scanf("%d", &x);
13    if (x > 0){
14        printf("X eh positivo\n");
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
```



## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- Dentro da opção **Debugging windows**, podemos habilitar a opção **Watches**.
  - Essa opção vai abrir uma pequena janela que permite ver o valor atual das variáveis de um programa, assim como o valor passado 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
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 }
```



## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- A partir de determinado ponto de pausa do programa, podemos nos mover para a próxima linha do programa com a opção **Next line (F7)**.
- Essa opção faz com que o programa seja executado passo a passo, sempre avançando para a linha seguinte do escopo onde estamos.

```
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
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 }
```

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- 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.
- Nesse caso, o triângulo amarelo que marca onde estamos no código vai para a primeira linha do código 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
```

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- Uma vez dentro de uma função, podemos percorrê-la passo a passo com a opção **Next line (F7)**.
  - Terminada a função, o **debugger** vai para a linha seguinte ao ponto do código que chamou a função (linha 16).
  - Caso queiramos ignorar o resto da função e voltar para onde estávamos no código que chamou a função, basta clicar na opção **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
```

## UTILIZANDO O DEBUGGER DO CODE::BLOCKS

- 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
```