

Programação 2020/2021 – 2º Semestre

Aula de Problemas 1

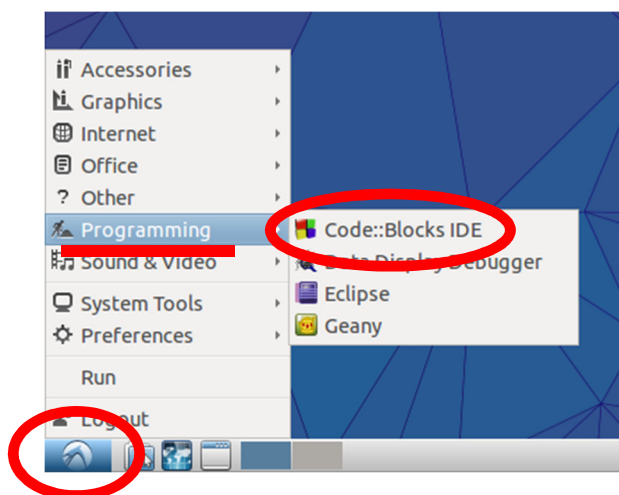
Os alunos deverão executar os passos aqui descritos no seu computador pessoal ou no laboratório. Durante a aula de problemas, o docente irá executar todos os passos necessários à implementação de um programa (que verifica que números são primos).

1. Code::Blocks – Criação de projectos

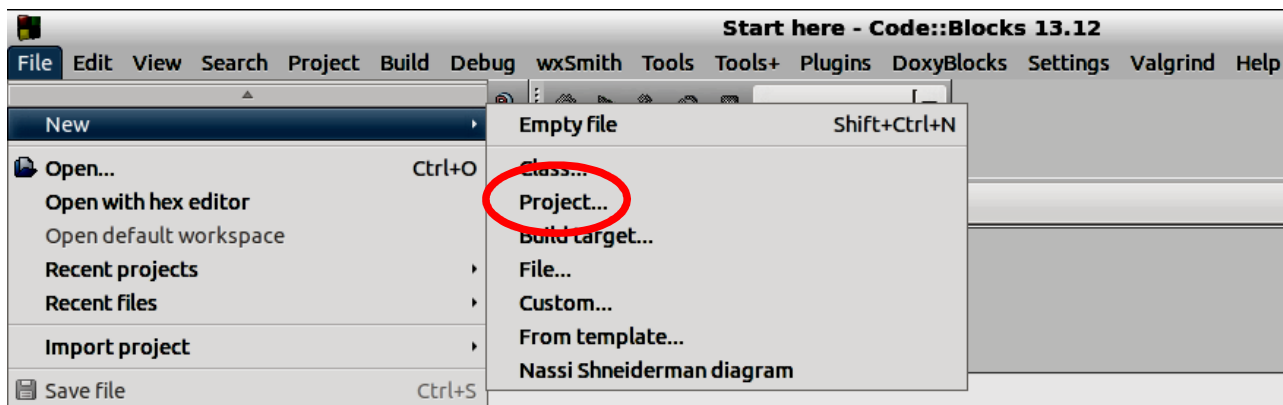
Para o desenvolvimento de aplicações em **C** iremos utilizar um ambiente de desenvolvimento integrado (IDE) designado por **Code::Blocks**.

Os vários ficheiros de código (.c e .h) que formarão uma aplicação são agrupados num projecto. Para criar um projecto é necessário seguir os seguintes passos:

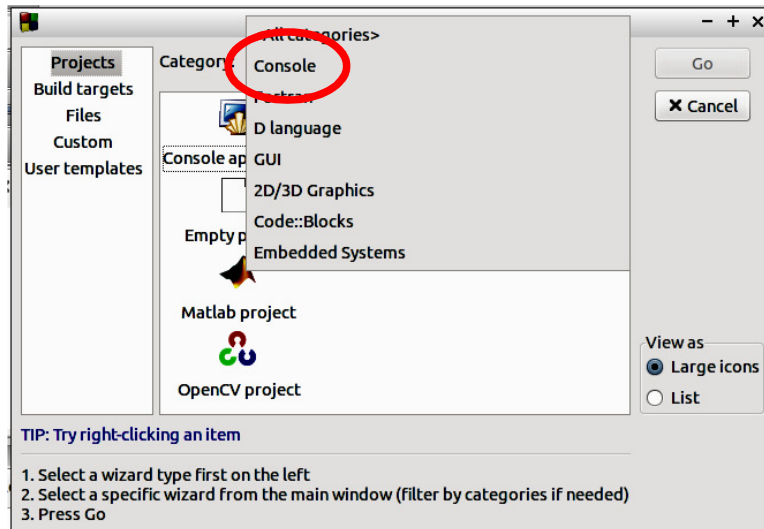
1. Abrir a aplicação **Code::Blocks**.



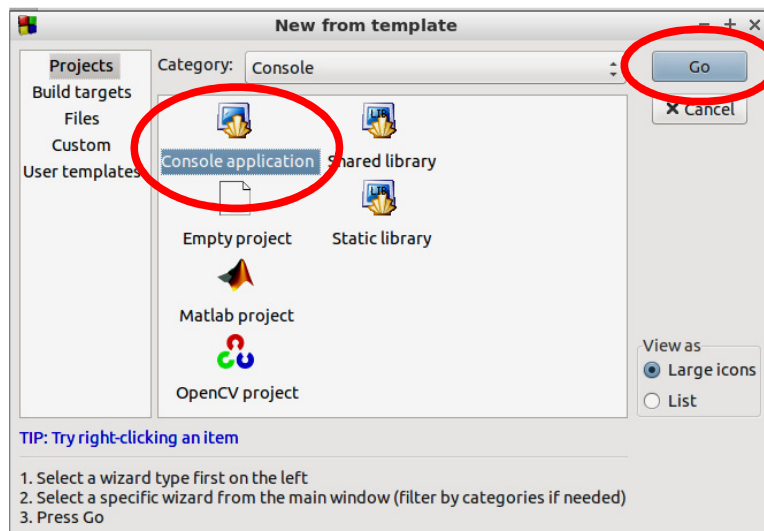
2. Seleccionar os menus **File** → **New** → **Project**.



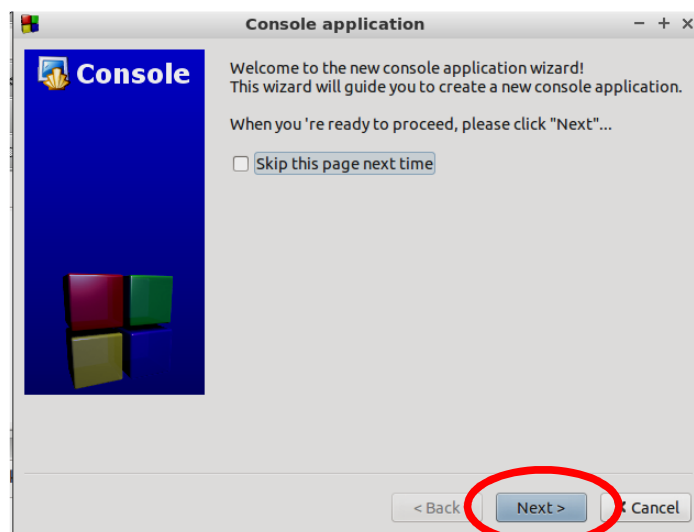
3. Seleccionar o tipo de aplicação a desenvolver (**Console**).



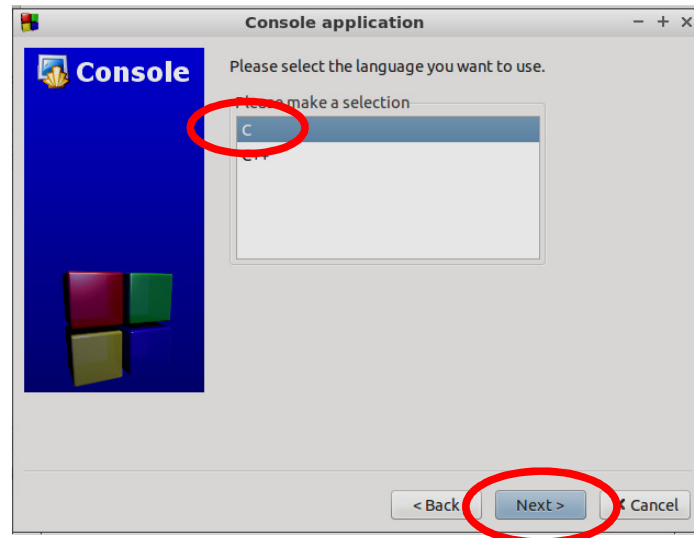
4. Seleccionar **Console Application** e carregar no botão **Go**.



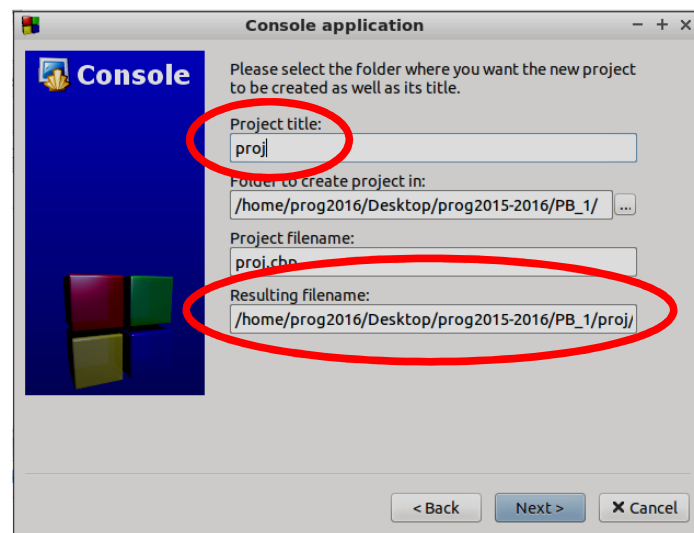
5. Carregar no botão **Next**.



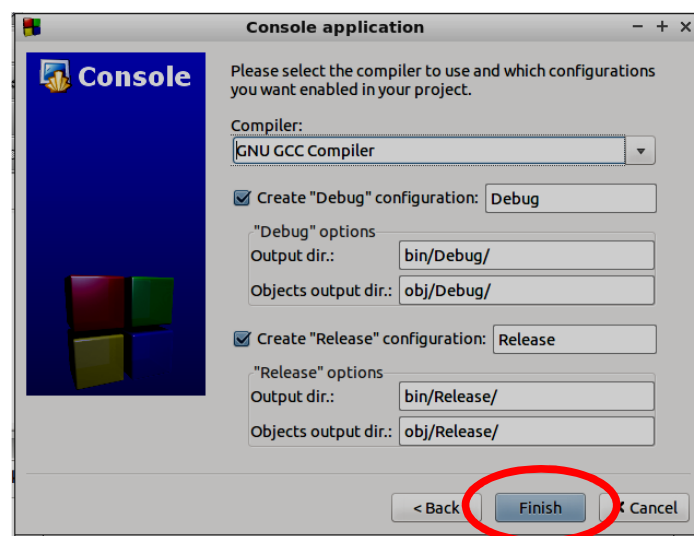
6. Seleccionar a linguagem de programação a usar no desenvolvimento da aplicação (C).



7. Na janela seguinte deverá ser introduzido o nome do projecto e a sua localização.



8. Finalizar a criação do projecto carregando no botão **Finish**.

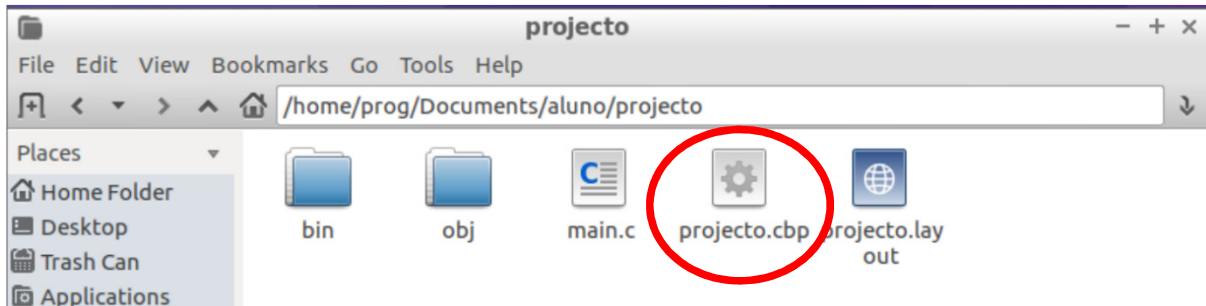


A partir deste momento é possível editar o código da aplicação, experimentá-la e verificar a sua correcção.

2. Code::Blocks – Abertura de projectos já existentes

Existem duas formas de abertura de um projecto já existente: 1) através do **File Manager** do Linux; ou 2) dentro do Code::Blocks.

1. Para abrir um projecto através do **File Manager** basta aceder à pasta do projecto e fazer duplo clique sobre o ficheiro com a extensão **.cbp**.

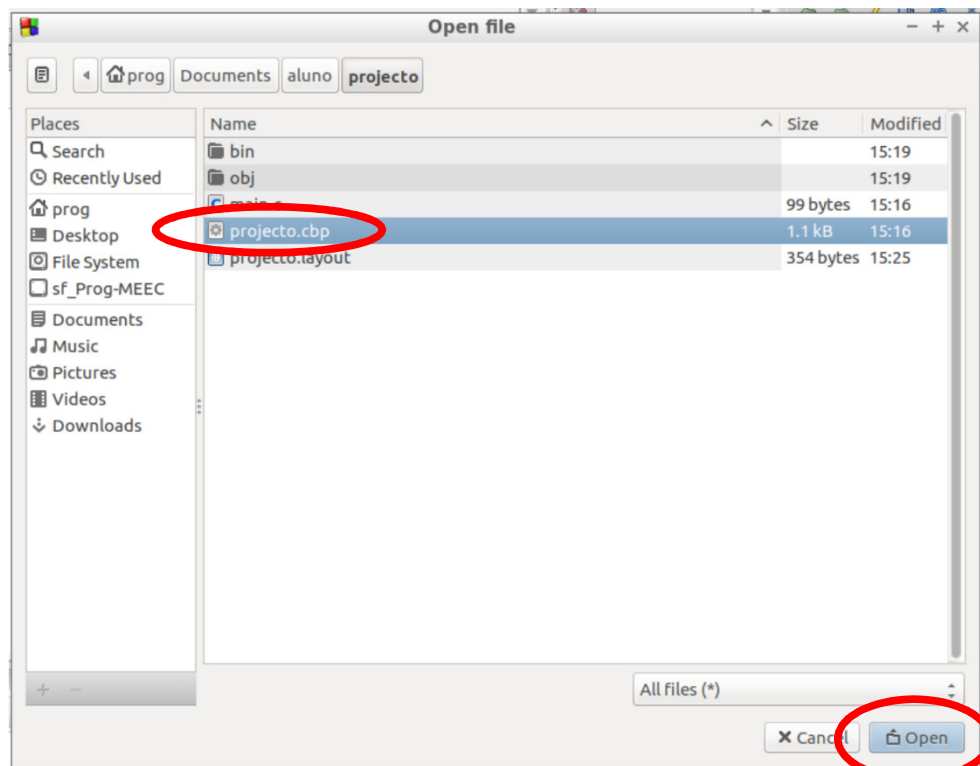


2. Em alternativa, dentro do Code::Blocks:

- Seleccionar os menus **File** → **Open**.



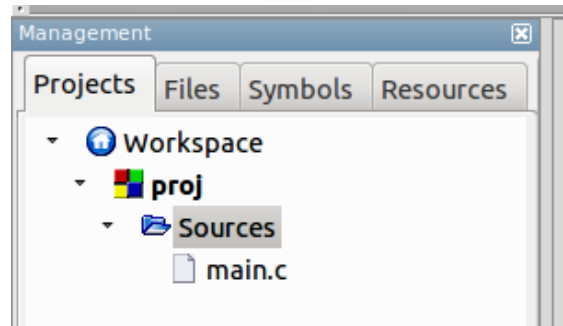
- Procurar a pasta correspondente ao projecto que se pretende abrir, seleccionar o ficheiro **.cbp** e carregar **Open**.



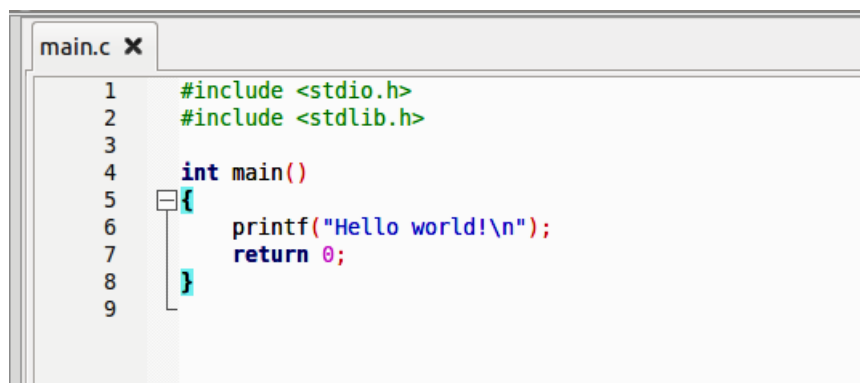
3. Code::Blocks – Funcionamento

O Code::Blocks integra várias funcionalidades: editor, compilador com indicação dos erros, ambiente de execução das aplicações desenvolvidas e depurador (para encontrar os erros de execução).









Do lado esquerdo da janela do Code::Blocks aparecem listados todos os ficheiros de código (.c e .h) do projecto. Na divisão **Sources** aparecerão os diversos ficheiros .c pertencentes ao projecto. Noutras divisões aparecerão os ficheiros .h.



Do lado direito da janela existe um editor de texto normal:

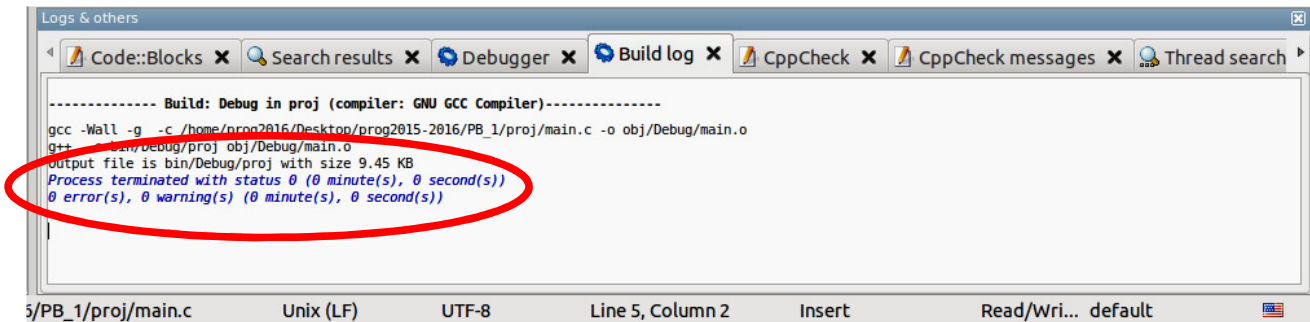


Para compilar e executar a aplicação existem botões que invocam o compilador com os parâmetros adequados:

 	Compila todos os ficheiros que foram alterados. Antes de compilar, grava todos os ficheiros pertencentes ao projecto que foram alterados.
 	Executa a aplicação. Não compila a aplicação, sendo executada a última versão compilada.
 	Compila todos os ficheiros que foram alterados e executa o programa gerado.
 	Cria todos os ficheiros auxiliares, e compila a aplicação.

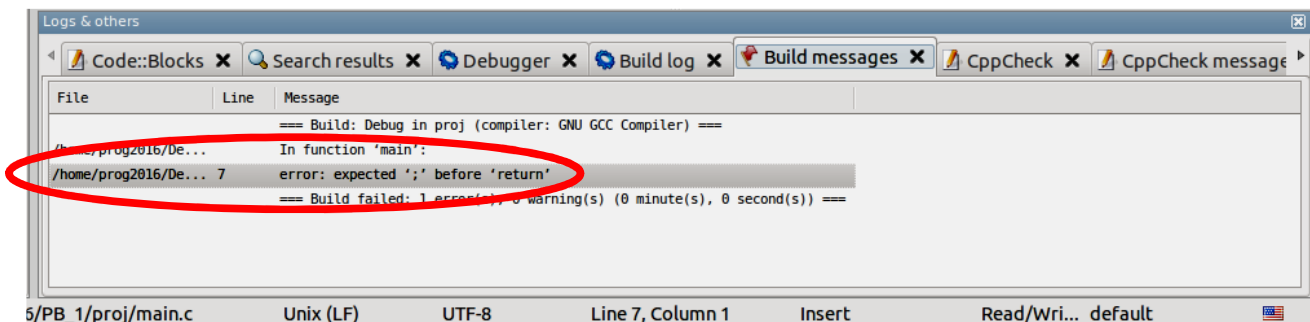
Os ícones poderão ter uma cor diferente, dependendo da versão do Linux instalada.

Se durante a compilação não forem detectados erros, no fundo da janela aparecerá uma mensagem semelhante à seguinte:



The screenshot shows the 'Build log' window in Code::Blocks. The message displayed is:
----- Build: Debug in proj (compiler: GNU GCC Compiler)-----
gcc -Wall -g -c /home/prog2016/Desktop/prog2015-2016/PB_1/proj/main.c -o obj/Debug/main.o
g++ -o bin/Debug/proj obj/Debug/main.o
Output file is bin/Debug/proj with size 9.45 KB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
The entire message is circled in red. The status bar at the bottom shows '5/PB_1/proj/main.c', 'Unix (LF)', 'UTF-8', 'Line 5, Column 2', 'Insert', 'Read/Wri... default'.

Caso o código tenha erros, estes serão apresentados.



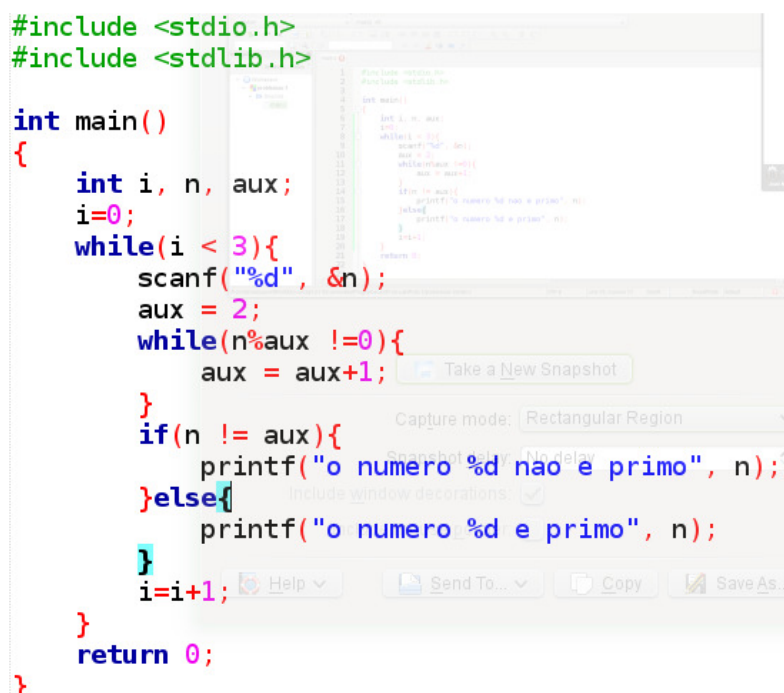
The screenshot shows the 'Build messages' window in Code::Blocks. The message displayed is:
==== Build: Debug in proj (compiler: GNU GCC Compiler) ====
In function 'main':
/home/prog2016/De... 7 error: expected ';' before 'return'
==== Build failed: 1 error(s); 0 warning(s) (0 minute(s), 0 second(s)) ====
The error message line is circled in red. The status bar at the bottom shows '5/PB_1/proj/main.c', 'Unix (LF)', 'UTF-8', 'Line 7, Column 1', 'Insert', 'Read/Wri... default'.

Clicando nessa linha, o cursor da janela de edição saltará para a linha com o erro.

4. Code::Blocks – Desenvolvimento de programas

Durante a aula prática, o docente desenvolverá um programa que lê 3 números inteiros e verifica quais deles são primos.

A solução final deverá ser próxima da seguinte, mas até chegar a essa solução é necessário desenvolver várias soluções parciais.



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

int main()
{
    int i, n, aux;
    i=0;
    while(i < 3){
        scanf("%d", &n);
        aux = 2;
        while(n%aux !=0){
            aux = aux+1;
        }
        if(n != aux){
            printf("o numero %d nao e primo", n);
        }else{
            printf("o numero %d e primo", n);
        }
        i=i+1;
    }
    return 0;
}
```

The screenshot shows a C program in Code::Blocks. The program reads three integers and checks if they are prime. The code is as follows:
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int i, n, aux;
 i=0;
 while(i < 3){
 scanf("%d", &n);
 aux = 2;
 while(n%aux !=0){
 aux = aux+1;
 }
 if(n != aux){
 printf("o numero %d nao e primo", n);
 }else{
 printf("o numero %d e primo", n);
 }
 i=i+1;
 }
 return 0;
}

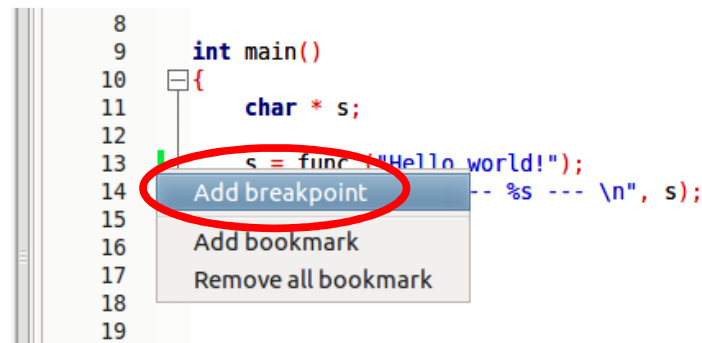
5. Code::Blocks – Depuração de programas

Copie o programa anterior e verifique se ele está totalmente correcto. Por exemplo, escreva no teclado valores negativos ou texto. Verifique o comportamento nestes casos.

Durante o semestre aprender-se-á a perceber e a resolver esses problemas.

Usando o *debugger* integrado no Code::Blocks, execute a aplicação passo a passo, de modo a visualizar o estado da aplicação e a variação dos valores das variáveis:

- Para marcar um *breakpoint* (local onde a execução da aplicação parará para observação do seu estado), é necessário carregar com o botão direito do rato na coluna contendo o número da linha.

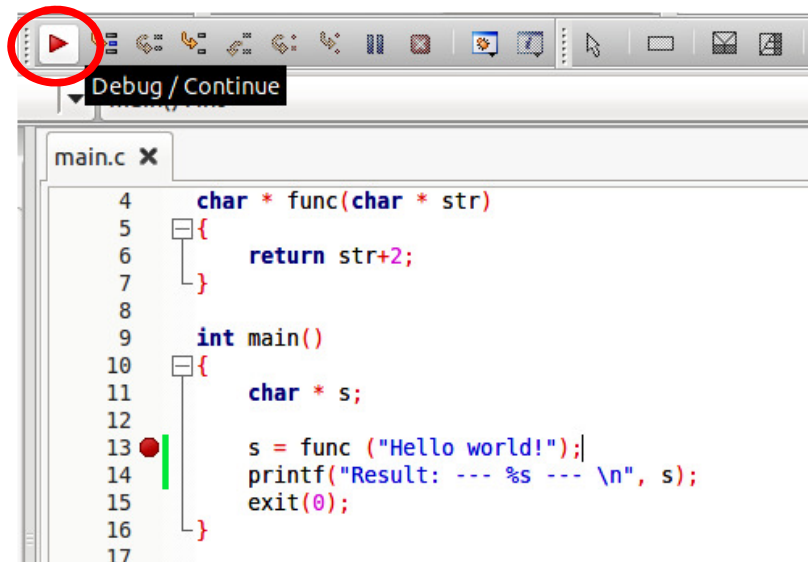


- Seleccionar a opção **Add breakpoint**. Aparecerá então uma marca nessa linha.



Pode-se então iniciar a aplicação dentro do *debugger*.

- Carregar no botão **Debug/Continue**.

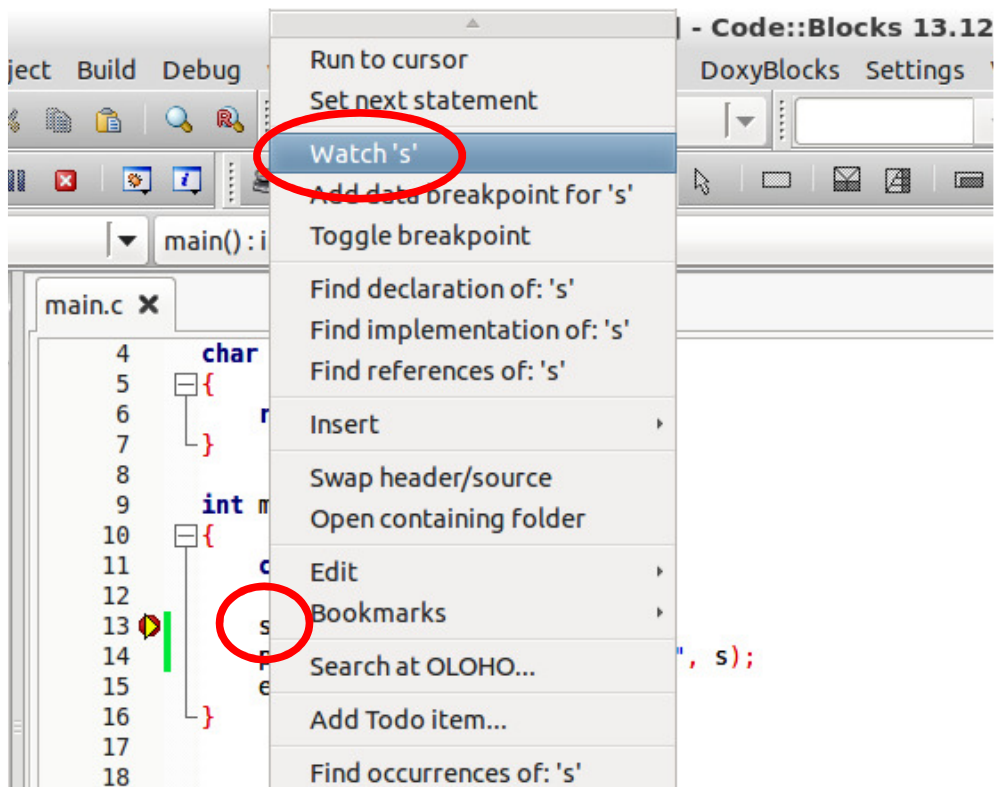


A aplicação executar-se-á até alcançar a linha onde foi inserido o *breakpoint*.

```
8
9
10 int main()
11 {
12     char * s;
13     s = func ("Hello world!");
14     printf("Result: --- %s --- \n", s);
15     exit(0);
16 }
17
```

O valor das diversas variáveis pode ser visto na janela de **Watches**. Para tal é necessário adicioná-las a essa janela:

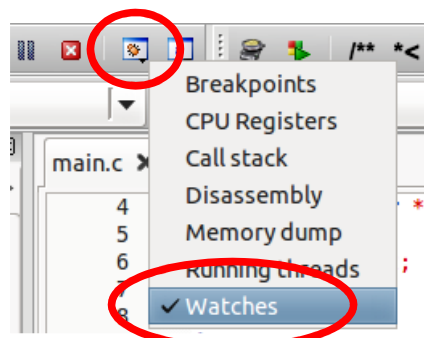
- Carregar com o botão direito do rato em cima da variável.



- Escolher a opção **Watch**.

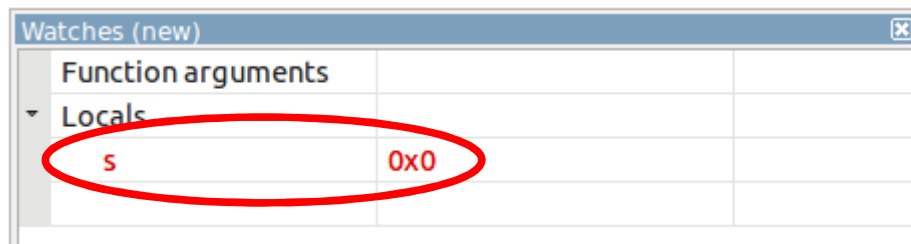
Se a janela **Watches** já estiver visível, o valor dessa variável aparecerá lá. Se a janela ainda não estiver visível é necessário abri-la. Para tal:

- Carregar no botão **Debugging Windows**.













- Activar a opção **Watches**.

O valor da variável pode então ser visualizado na janela **Watches**:



Para continuar a execução da aplicação podem-se usar os seguintes botões:

		Debug/Continue	Continua a execução da aplicação até terminar ou encontrar um <i>breakpoint</i> .
		Next Line	Executa a linha actual, parando na linha seguinte e "saltando por cima de funções".
		Step into	Executa a linha actual, parando numa linha de código diferente. Se a linha actual for uma função, entra e para na sua primeira linha de código.
		Step out	Continua a execução do código, parando apenas quando sair da função actual.
		Stop debugger	Termina a execução da aplicação.



Erros de compilação

Depois de experimentar o programa, introduza erros sintácticos, mas um de cada vez:

- Apague um ponto e vírgula.
- Apague uma das primeiras linhas.
- Apague a palavra *main*.
- Introduza texto no meio do código.
- Apague uma qualquer chaveta.
- ...