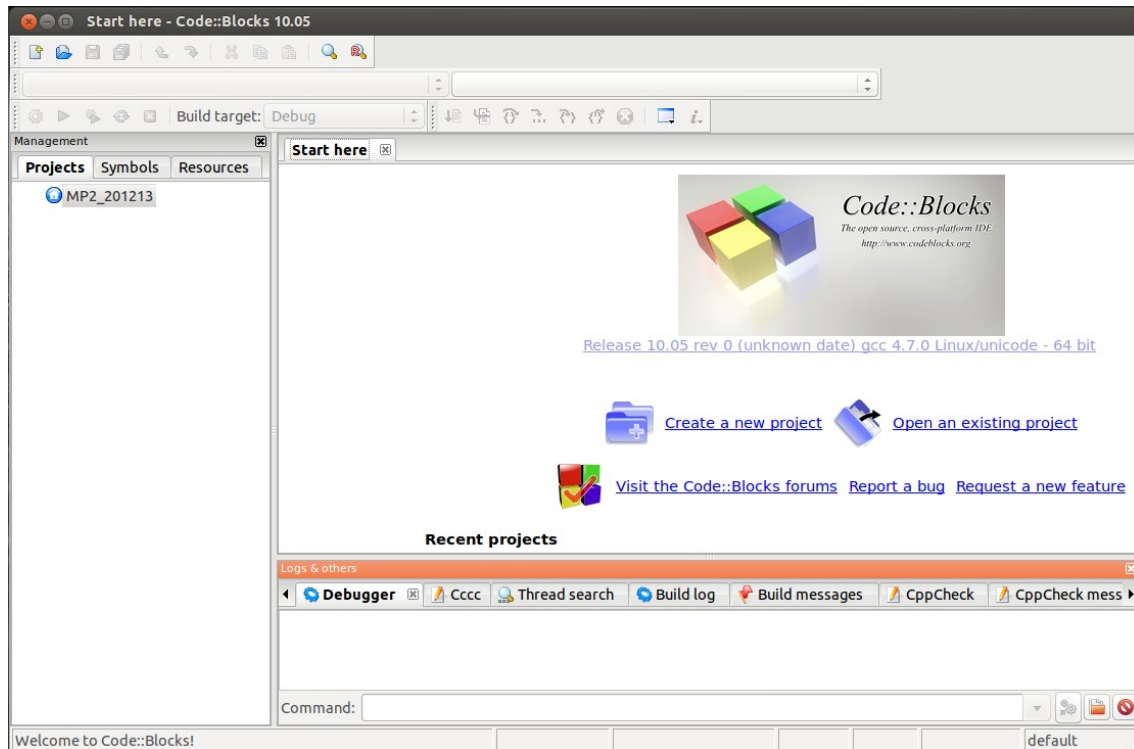


Tutorial sobre utilização do IDE Code::Blocks para compilar e depurar programas em C

A maioria das figuras refere-se à versão Code::Blocks 10.05 em Ubuntu © António Esteves

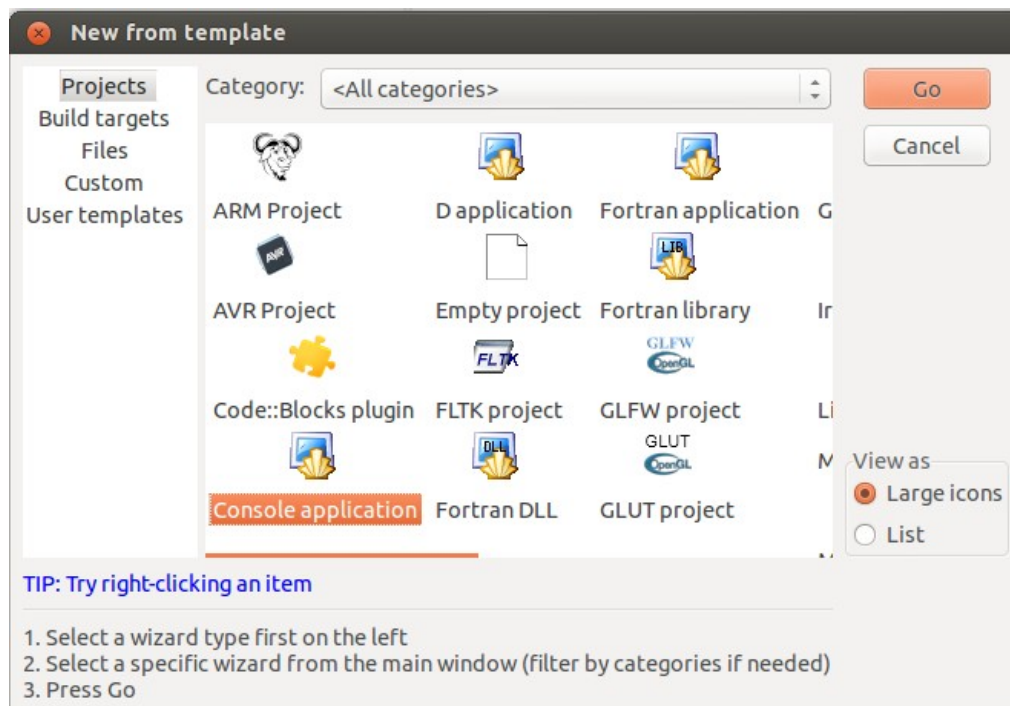
Correr a aplicação **Code::Blocks**.

Mudar o nome do **Workspace** para "MP2_201516" (as figuras do tutorial utilizam "MP2_201213").



Criar um novo projeto clicando no link "Create new Project" no separador "Start here".

Escolha o tipo de projeto "Console application".

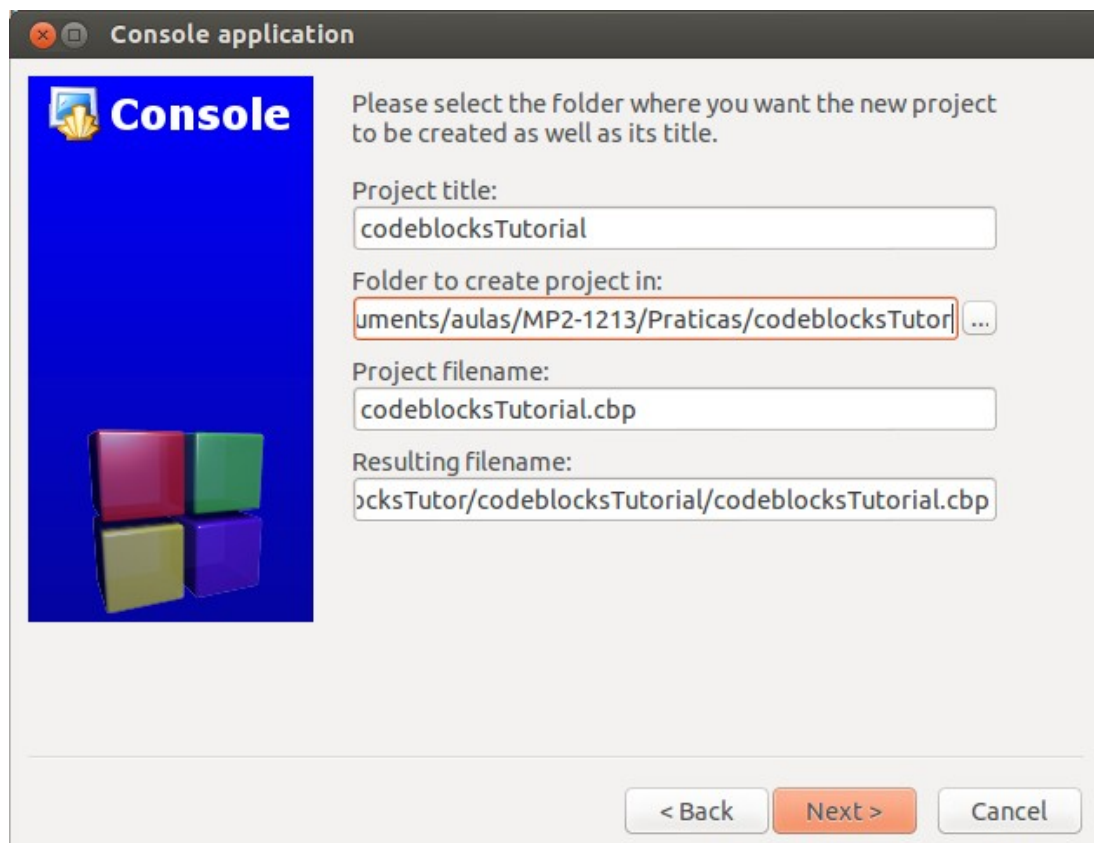


Clicar em "Next" na janela seguinte.

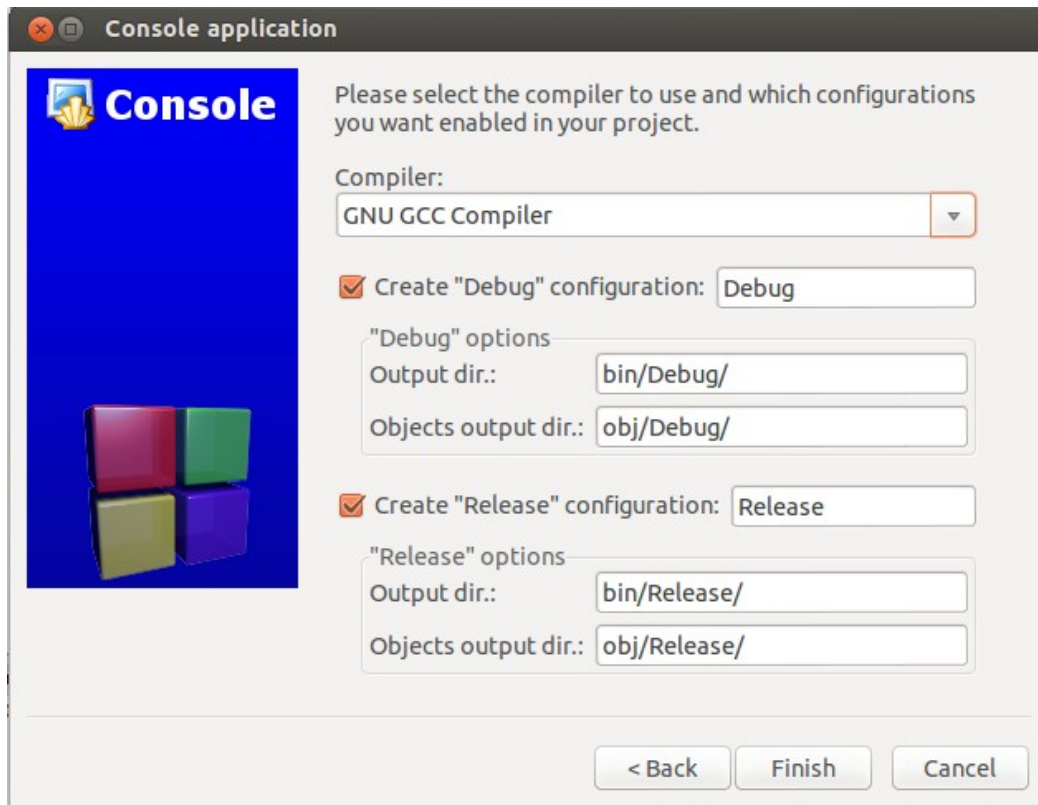
Escolher a linguagem "C".



Escolher o nome do projeto e a pasta onde vai ficar guardado.

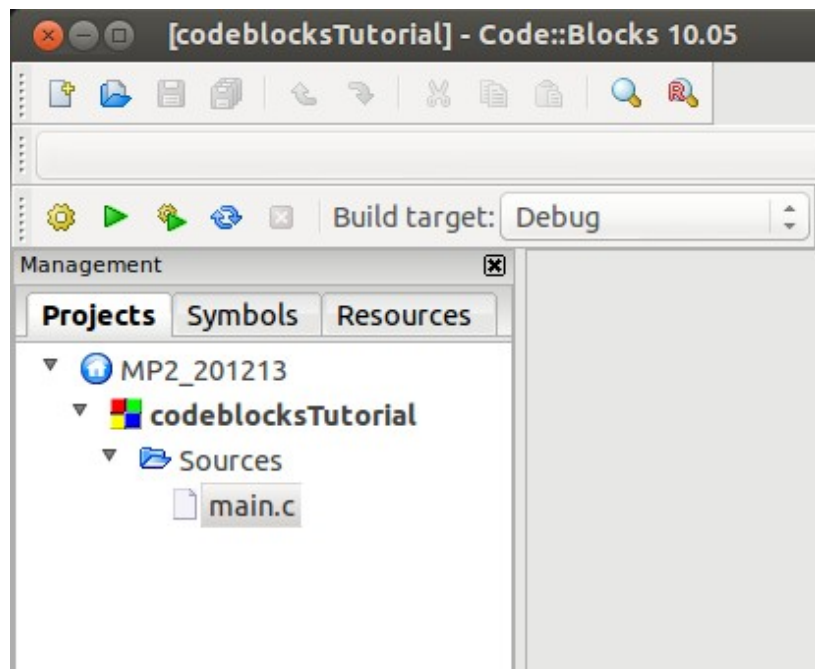


Escolher como compilador a usar "GNU GCC Compiler"



Clicar em "Finish".

O Code:Blocks cria automaticamente um ficheiro "**main.c**" no projeto, o qual podemos usar para começar a inserir o nosso código.



Abrir o ficheiro "**main.c**" e introduza o seguinte código C:

```
int main() // Calcula o dia da semana correspondente a uma data dd/mm/aaaa
{
    int meses, dia, ano;
    printf("Introduza uma data para a qual deseja saber\n");
    printf(" o dia da semana (DD MM AAAA)? ");
    scanf("%d%d%d", &dia, &mes);
    if (ano < 1752)
        printf("So' sao aceites datas posteriores a 1752\n");
    else {
        if (mes < 3) { // Jan e Fev = mes 13 e 14 do ano precedente
            mes += 12;
            ano -= 1;
        } // FIM DO if
        diasemana = (dia + 2*mes + 3*(mes+1)/5 + ano + ano/4 - ano/100 + ano/400 + 1) % 7;
        if (mes > 12) { // reset Jan e Fev
            mes -= 12;
            ano += 1;
        } // FIM DO if
        printf("O dia %d/%d/%d ocorre num(a) ", dia, mes, ano);
        switch (diasemana)
        {
            case 0: printf("Domingo\n"); break;
            case 1: printf("Segunda\n"); break;
            case 2: printf("Terça\n"); break;
            case 3: printf("Quarta\n"); break;
            case 4: printf("Quinta\n"); break;
            case 5: printf("Sexta\n"); break;
            case 6: printf("Sabado\n"); break;
        } // FIM DO switch
    } // FIM DO else
    return 0;
} // FIM DO main
```

NOTA: Se copiar o código do enunciado em formato PDF e em Ubuntu, remova as linhas em branco e substitua cada caractere menos ("-" a preto) por um novo caractere menos ("-") mas que ficará assinalado a vermelho.

Depois de escrever o código C, convém guardar o ficheiro.

Vamos agora construir (*build*) o projeto. O processo de construção envolve as fases de compilação e ligação (*link*), de modo a gerar um ficheiro executável.

Tente construir o projeto, clicando no botão de **"Build"**.

Observe as mensagens de erro e de aviso que resultaram da compilação, apresentadas na janela inferior esquerda do IDE, no separador **"Build messages"**. Se esta janela não estiver visível, abra-a indo ao menu **"View"** e selecionando **"Logs"**.

The screenshot shows the Code::Blocks IDE with a C program in `main.c` and its compilation errors in the `Logs & others` window.

main.c:

```

1  int main() // Calcula o dia da semana correspondente a uma data dd/mm/aaaa
2  {
3      int meses, dia, ano;
4      printf("Introduza uma data para a qual deseja saber\n");
5      printf(" o dia da semana (DD MM AAAA)? ");
6      scanf("%d%d%d", &dia, &mes);
7      if (ano < 1752)
8          printf("So' sao aceites datas posteriores a 1752\n");
9      else {
10         if (mes < 3) { // Jan e Fev = mes 13 e 14 do ano precedente
11             mes += 12;
12         }

```

Logs & others:

File	Line	Message
=== codeblocksTutorial, Debug ===		
/home/esteve...		In function 'main':
/home/esteve...	5	warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
/home/esteve...	5	warning: incompatible implicit declaration of built-in function 'printf' [enabled by default]
/home/esteve...	7	warning: implicit declaration of function 'scanf' [-Wimplicit-function-declaration]
/home/esteve...	7	warning: incompatible implicit declaration of built-in function 'scanf' [enabled by default]
/home/esteve...	7	error: 'mes' undeclared (first use in this function)
/home/esteve...	7	note: each undeclared identifier is reported only once for each function it appears in
/home/esteve...	15	error: 'diasemana' undeclared (first use in this function)
/home/esteve...	21	error: 'diaSemana' undeclared (first use in this function)
/home/esteve...	33	error: expected declaration or statement at end of input
/home/esteve...	33	error: expected declaration or statement at end of input
/home/esteve...	4	warning: unused variable 'meses' [-Wunused-variable]
/home/esteve...	33	warning: control reaches end of non-void function [-Wreturn-type]
=== Build finished: 5 errors, 6 warnings (0 minutes, 0 seconds) ===		

Clique numa mensagem de erro, como por exemplo no erro assinalado na próxima figura como ocorrendo na linha 15, para que essa linha seja assinalada na janela do código C.

The screenshot shows the Code::Blocks IDE with the same C program in `main.c` and its compilation errors in the `Logs & others` window. Line 15 in the code is highlighted, corresponding to the error message in the logs.

main.c:

```

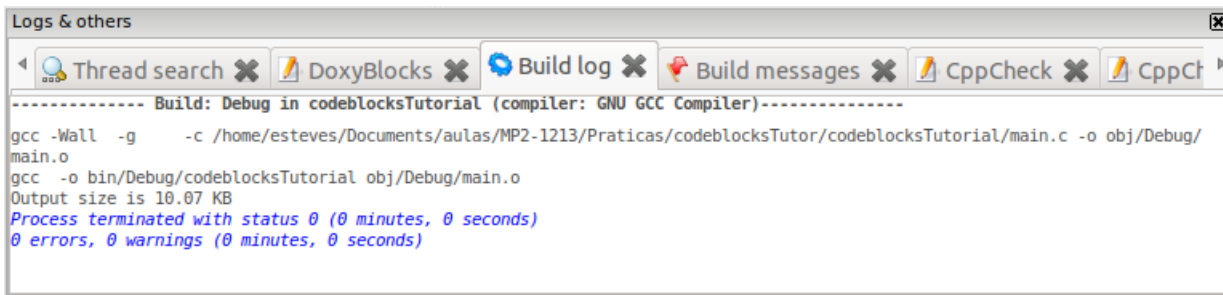
1  int main() // Calcula o dia da semana correspondente a uma data dd/mm/aaaa
2  {
3      int meses, dia, ano;
4      printf("Introduza uma data para a qual deseja saber\n");
5      printf(" o dia da semana (DD MM AAAA)? ");
6      scanf("%d%d%d", &dia, &mes);
7      if (ano < 1752)
8          printf("So' sao aceites datas posteriores a 1752\n");
9      else {
10         if (mes < 3) { // Jan e Fev = mes 13 e 14 do ano precedente
11             mes += 12;
12             ano -= 1;
13         } // FIM DO if
14         diasemana = (dia + 2*mes + 3*(mes+1)/5 + ano + ano/4 - ano/100 + ano/400 + 1) % 7;
15         if (mes > 12) { // reset Jan e Fev
16             mes -= 12;
17             ano += 1;
18         } // FIM DO if
19         printf("O dia %d/%d/%d ocorre num(a) ", dia, mes, ano);
20         switch (diasemana)
21         {

```

Logs & others:

File	Line	Message
/home/esteve...	7	error: 'mes' undeclared (first use in this function)
/home/esteve...	7	note: each undeclared identifier is reported only once for each funct
/home/esteve...	15	error: 'diasemana' undeclared (first use in this function)
/home/esteve...	21	error: 'diaSemana' undeclared (first use in this function)
/home/esteve...	33	error: expected declaration or statement at end of input
/home/esteve...	33	error: expected declaration or statement at end of input

Corrija agora os erros e avisos restantes e construa o projeto até não haver mensagens de erro ou de aviso. Veja quais os comandos usados na compilação e "linking", apresentados na janela inferior esquerda do IDE, no separador **"Build log"**.

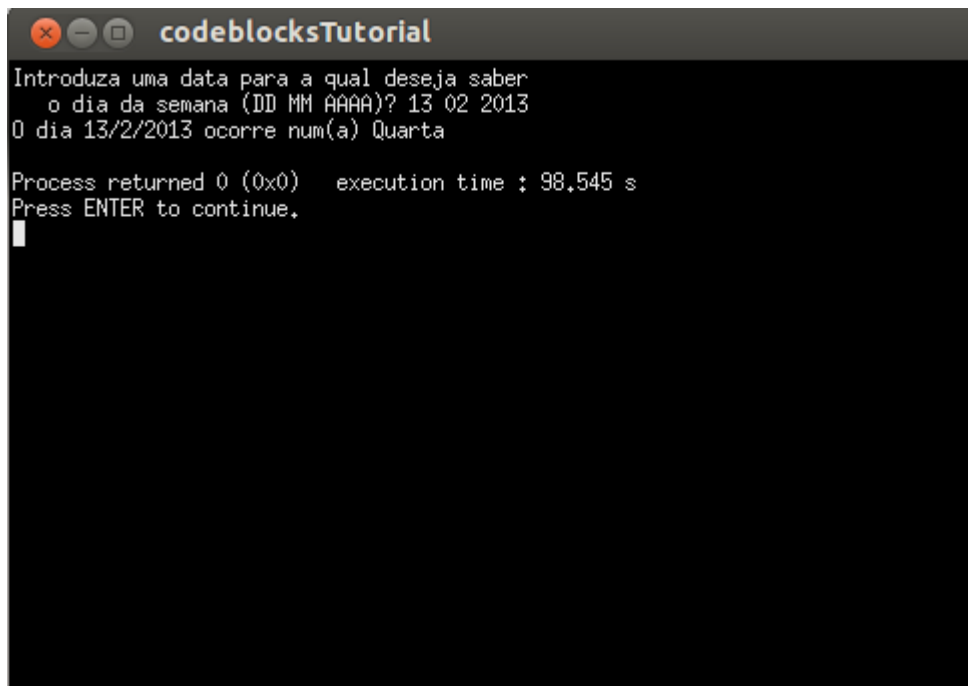


```
----- Build: Debug in codeblocksTutorial (compiler: GNU GCC Compiler)-----
gcc -Wall -g -c /home/esteves/Documents/aulas/MP2-1213/Praticas/codeblocksTutor/codeblocksTutorial/main.c -o obj/Debug/main.o
gcc -o bin/Debug/codeblocksTutorial obj/Debug/main.o
Output size is 10.07 KB
Process terminated with status 0 (0 minutes, 0 seconds)
0 errors, 0 warnings (0 minutes, 0 seconds)
```

Depois de ter compilado o programa "**main.c**" com sucesso, de o ter ligado à biblioteca necessária (**stdio.h**) e gerado um programa executável, é altura de o executar.

Para isso pode clicar-se no botão de "**Run**".

Nesta altura deve ter uma consola com o programa em execução semelhante à seguinte.

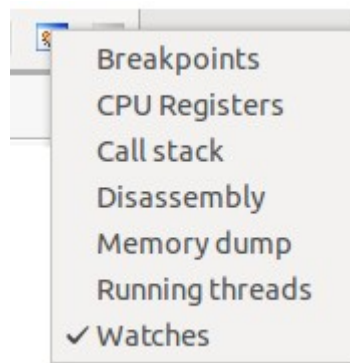


```
codeblocksTutorial
Introduza uma data para a qual deseja saber
o dia da semana (DD MM AAAA)? 13 02 2013
O dia 13/2/2013 ocorre num(a) Quarta

Process returned 0 (0x0)   execution time : 98.545 s
Press ENTER to continue.
```


- Se estivermos a depurar as instruções duma função, terminar a execução dessa função.
- Executar a próxima instrução *assembly* (⇔ execução passo-a-passo ao nível do *assembly*)
- Se a próxima instrução *assembly* for a chamada de uma função (CALL), começar a executar a primeira instrução dessa função
- Fazer “uma pausa” na depuração
- Terminar a depuração
- Selecionar as janelas, auxiliares da depuração, que se pretende abrir
- Ver informação diversa (funcionalidades avançadas, não pertinentes neste tutorial).

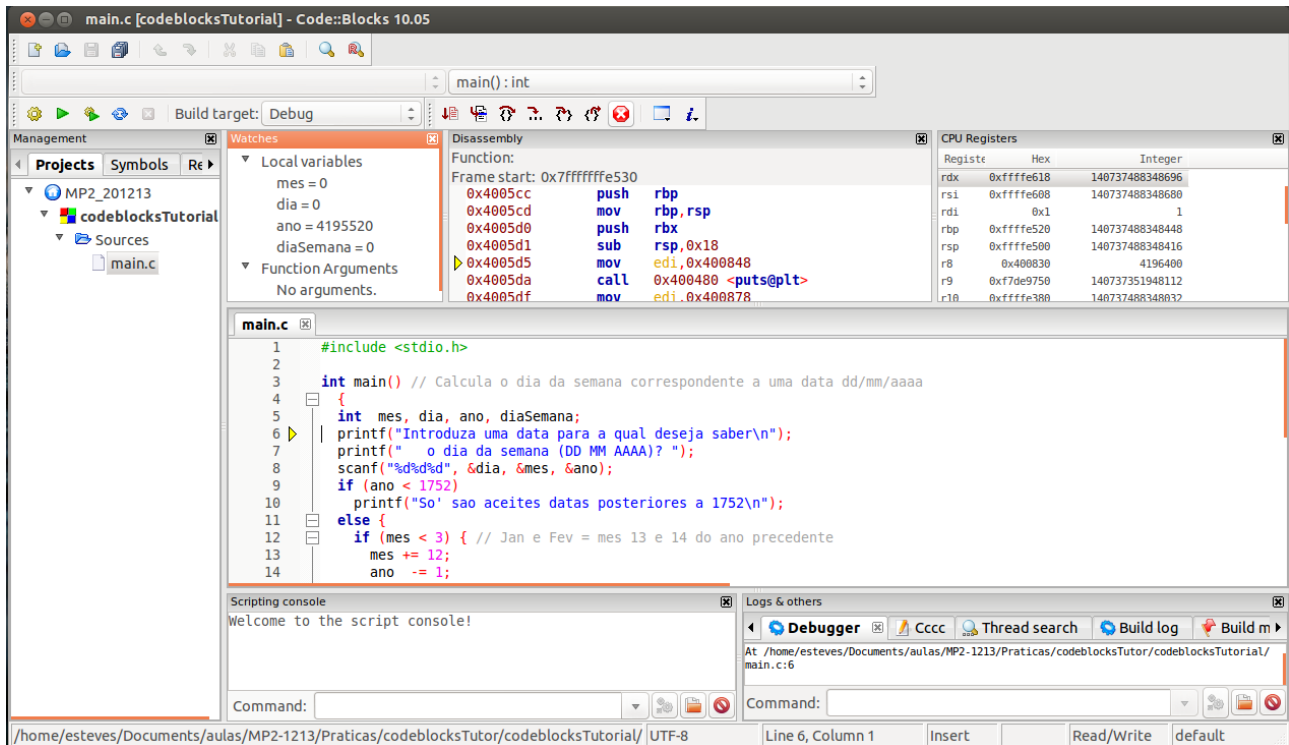
Ao clicar no botão “**Debugging Windows**” podemos escolher as seguintes janelas auxiliares da depuração:



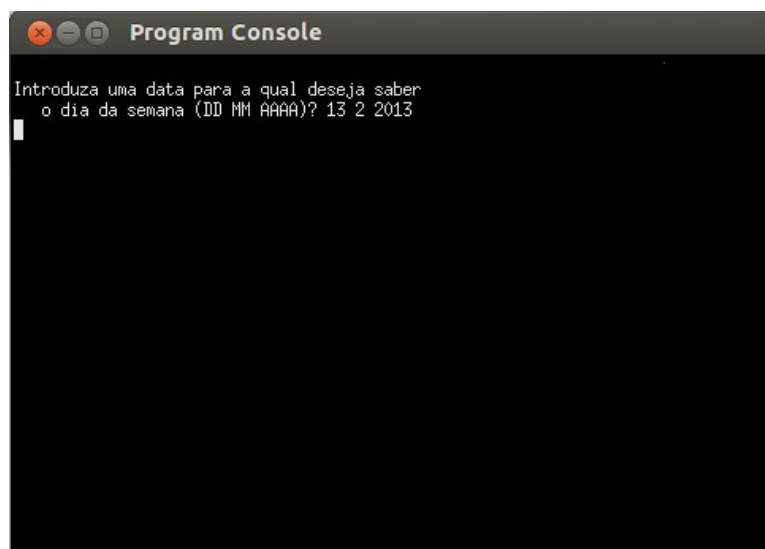
- Pontos de paragens definidos (“**Breakpoints**”)
- Registos do processador (“**CPU Registers**”)
- Pilha com as funções chamadas (“**Call stack**”)
- Instruções *assembly* correspondentes ao código em execução (“**Disassembly**”)
- Conteúdo da memória (“**Memory dump**”)
- Fios de execução (“**Running threads**”)
- Variáveis locais, argumentos da função invocada e outras expressões a inspecionar (“**Watches**”).

Selecione as janelas para visualizar os registos do processador, as instruções *assembly* e “**Watches**”.

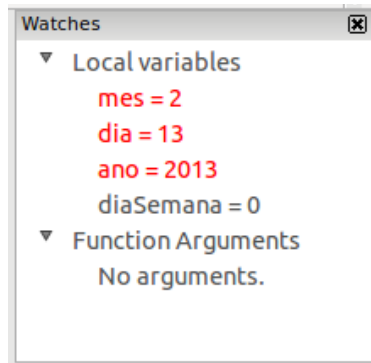
Coloque o cursor no primeiro `printf()` do `main()` e clique no botão “**Run to cursor**”. O estado da depuração será semelhante ao apresentado na próxima figura.



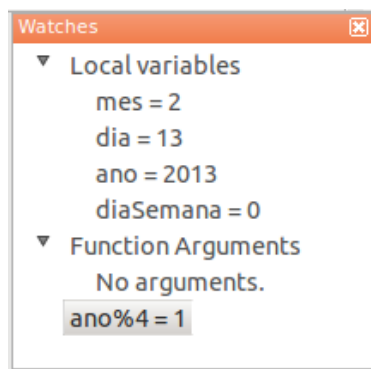
Para continuar a execução passo-a-passo, clicar repetidamente no botão “Next line”. Cada clique faz com que uma instrução/linha em C seja executada. Repita a execução passo-a-passo até atingir a linha com “`if(ano < 1752)`”. Quando a execução atinge uma linha que exige a introdução de dados a partir do teclado (por exemplo, a linha com `scanf()`), é preciso fazê-lo na consola em que o programa está a ser executado. Além da introdução de dados, a consola de execução do programa também serve para visualizar os dados de saída do programa.



Nesta fase é possível observar na janela “Watches” se as variáveis (`dia`, `mes`, `ano`) possuem os valores introduzidos a partir do teclado. Os valores ficam em cor vermelha quando mudaram na última linha/instrução executada.

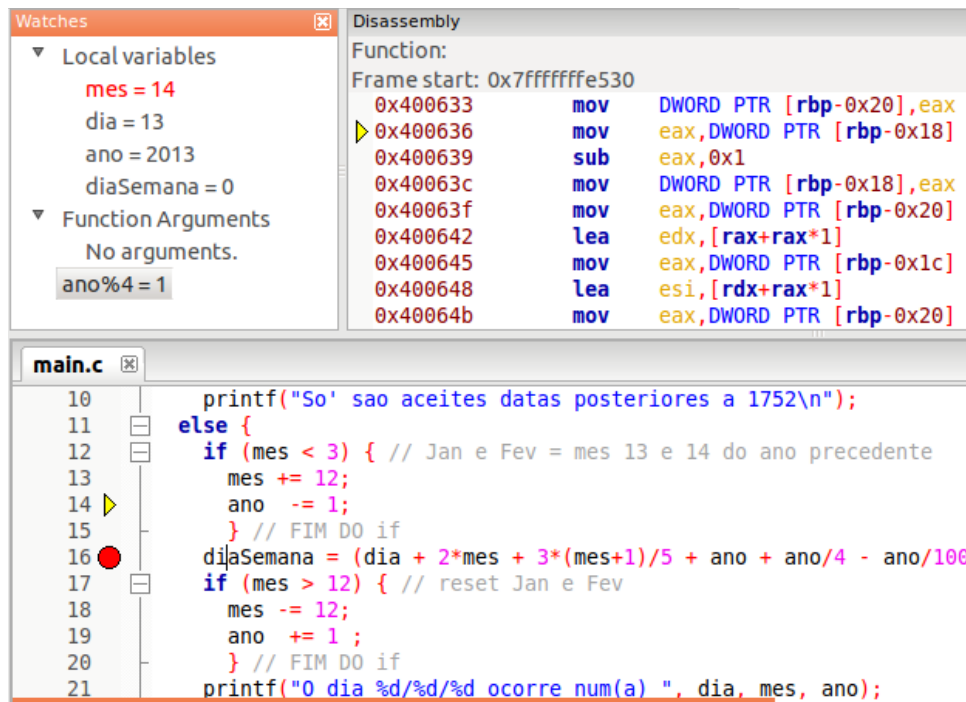


A janela “**Watches**” permite ainda analisar expressões, enquanto se faz a depuração do programa. Para isso clique com o rato dentro desta janela, na última linha que está vazia, e introduza por exemplo a expressão “*ano%4*”.



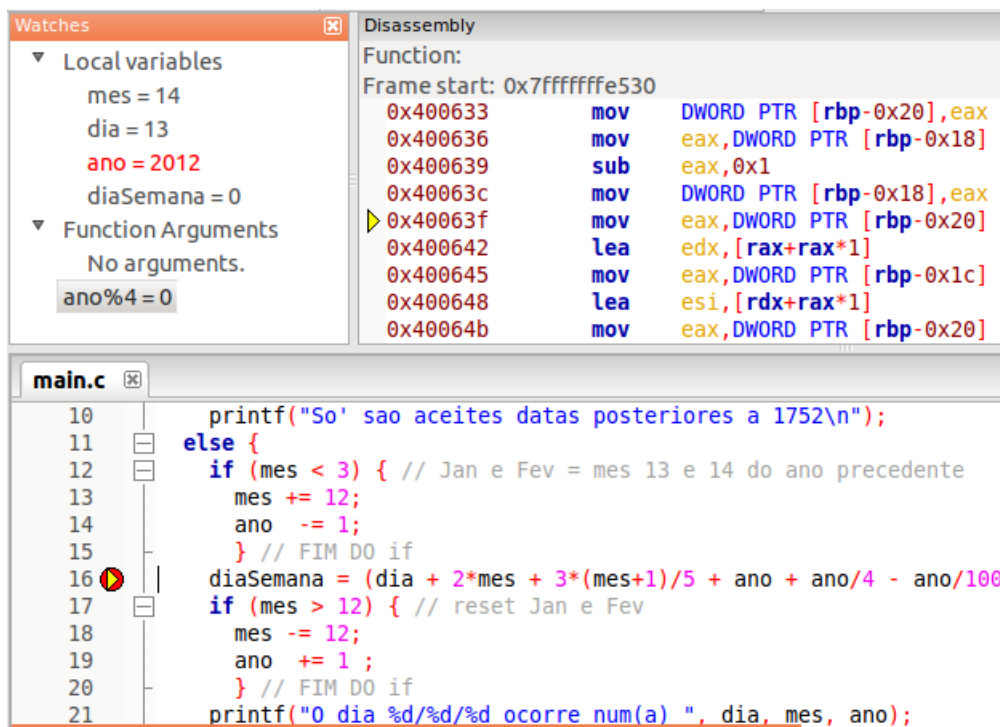
Definir Pontos de Paragem Incondicionais

Para definir um ponto de paragem incondicional, coloque o cursor na linha do código em que deseja parar a execução. Depois, marque o ponto de paragem usando a tecla **F5** ou o menu “**Debug → Toggle breakpoint**”. Nesta altura deverá observar um grande ponto vermelho junto da linha selecionada.

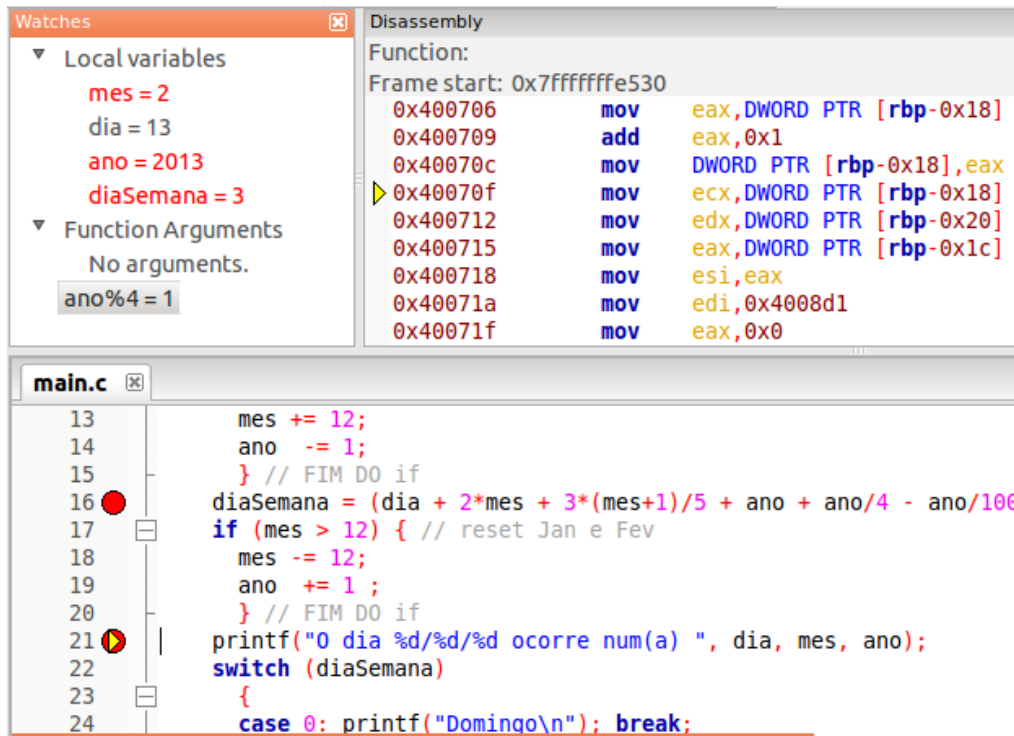


Executar até um Ponto de Paragem

Para executar o programa até ao ponto de paragem marcado, clique no botão “**Debug/Continue**” ou prima **CTL+F7**. Esta ação faz com que o programa seja executado até ao próximo ponto de paragem.



Marque outro ponto de paragem mais abaixo no código (`printf()`) antes do `switch()` e repita o procedimento para executar o programa até ao novo ponto de paragem.

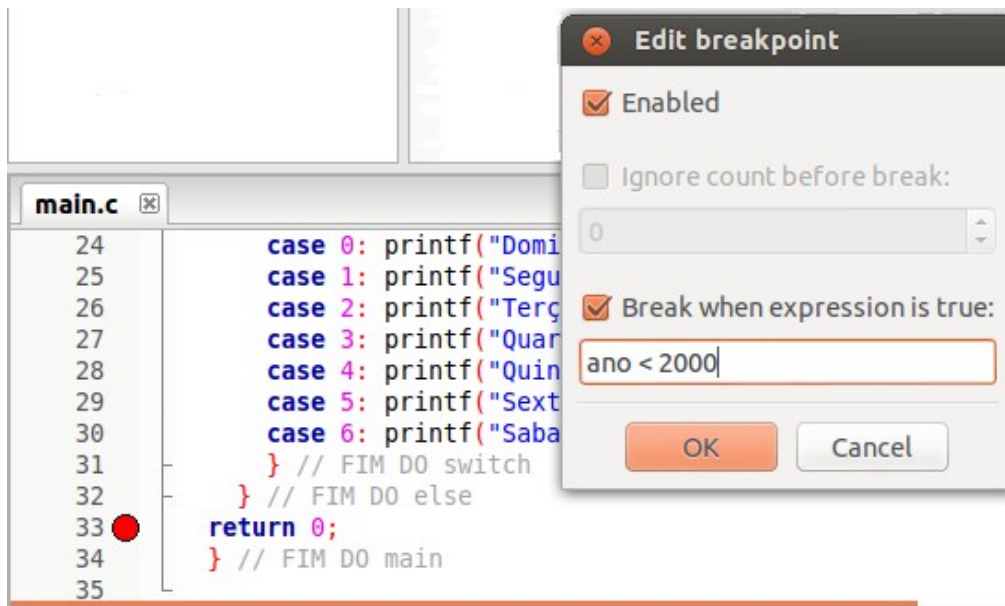


Para remover um ponto de paragem, coloque-se na linha respetiva e prima **F5** ou posicione o rato sobre o ponto vermelho e prima o botão esquerdo. Remova o ponto assinalado na linha 16 na figura anterior.

Pontos de Paragem Condicionais

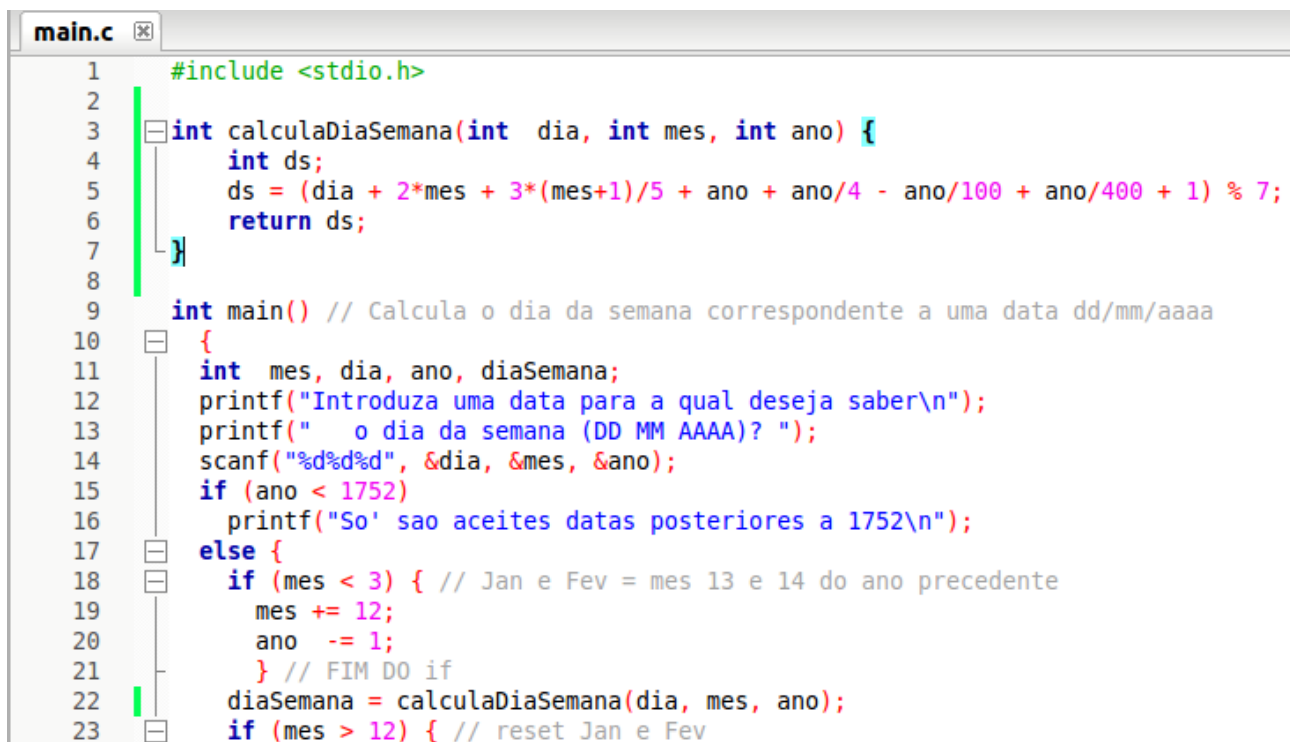
Um ponto de paragem condicional só faz parar a execução do programa se uma determinada condição for verdadeira. Os pontos de paragem condicionais são normalmente usados para parar a execução dum programa quando se está a executar um ciclo. Para marcar um ponto de paragem condicional, posicione o rato sobre o ponto vermelho respetivo, prima o botão direito e selecione **"Edit breakpoint"**.

Crie um ponto de paragem na linha com **"return"** e marque-o como sendo condicional. A próxima figura mostra um exemplo em que se define um ponto de paragem em que a execução só vai parar se o "ano for menor que 2000".



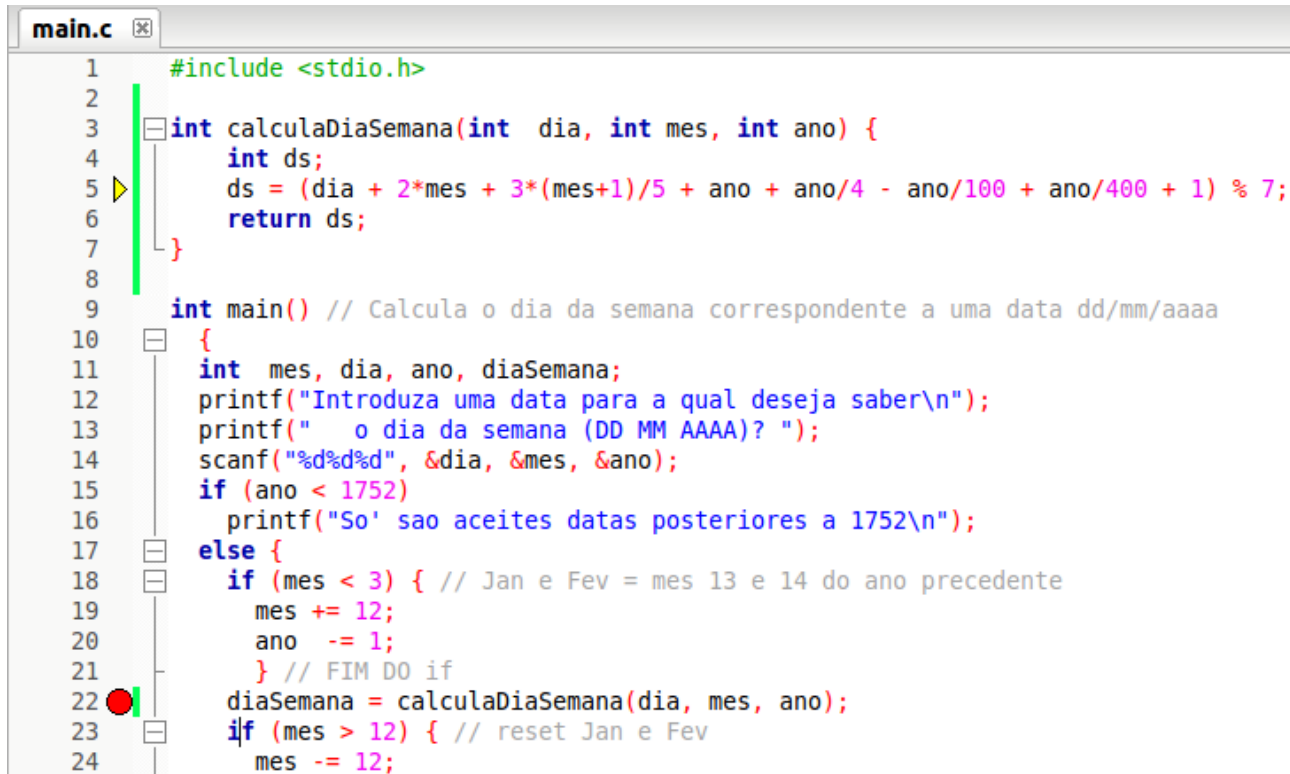
Exercite as potencialidades do depurador, introduzindo vários pontos de paragem e executando o programa com várias datas.

Para exercitar as funcionalidades “**Step into**” e “**Step out**” vamos alterar o código para introduzir uma função que calcula o dia da semana: `int calculaDiaSemana(int dia, int mes, int ano)`. O código desta função é o indicado nas linhas 3 a 7 da próxima figura. Esta função será utilizada para calcular o valor a atribuir à variável `diaSemana`, como se mostra na linha 22 da mesma figura.



Depois de alterado o código, fazemos “**Build**” do projeto. Se não houver erros, podemos fazer uma nova depuração. Para isso, introduzimos um ponto de paragem na linha onde é chamada a função

`calculaDiaSemana()`. Quando a execução atingir este ponto de paragem, clicamos no botão “**Step into**”. Deste modo, em vez de ser executado todo o código da função e parar na linha seguinte do `main()` (é o que acontecia se clicássemos em “**Next line**”), a execução salta para a primeira instrução da função (ver próxima figura).

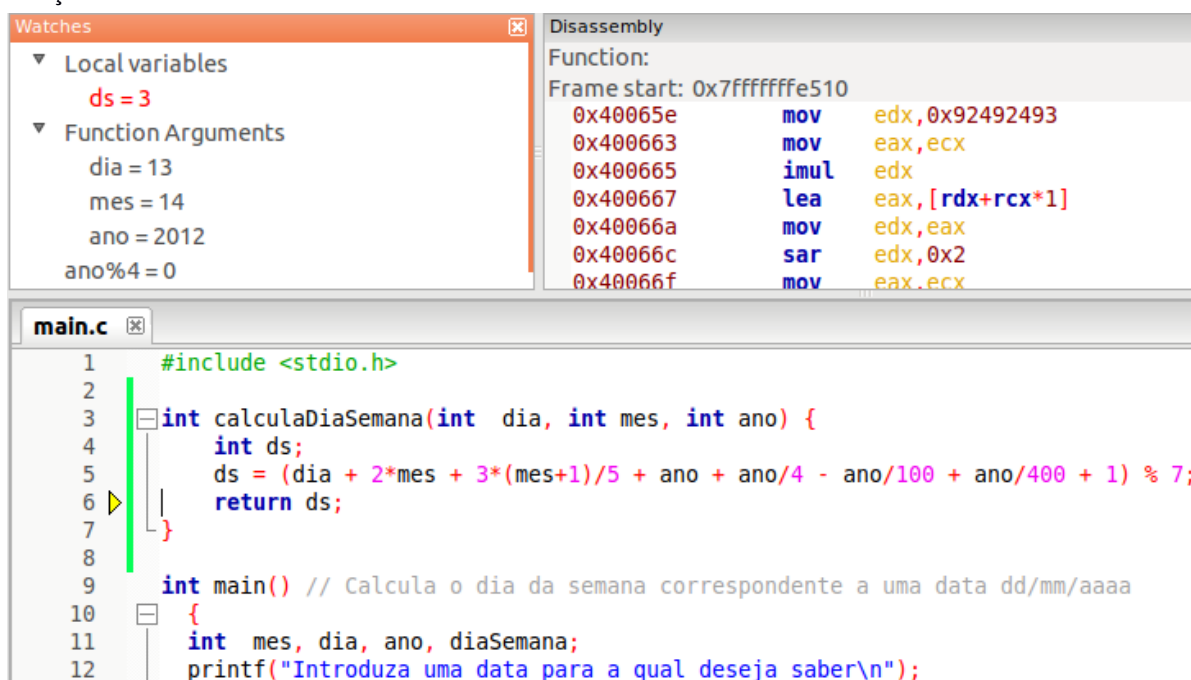


```

1  #include <stdio.h>
2
3  int calculaDiaSemana(int dia, int mes, int ano) {
4      int ds;
5      ds = (dia + 2*mes + 3*(mes+1)/5 + ano + ano/4 - ano/100 + ano/400 + 1) % 7;
6      return ds;
7  }
8
9  int main() // Calcula o dia da semana correspondente a uma data dd/mm/aaaa
10 {
11     int mes, dia, ano, diaSemana;
12     printf("Introduza uma data para a qual deseja saber\n");
13     printf("    o dia da semana (DD MM AAAA)? ");
14     scanf("%d%d%d", &dia, &mes, &ano);
15     if (ano < 1752)
16         printf("So' sao aceites datas posteriores a 1752\n");
17     else {
18         if (mes < 3) { // Jan e Fev = mes 13 e 14 do ano precedente
19             mes += 12;
20             ano -= 1;
21         } // FIM DO if
22         diaSemana = calculaDiaSemana(dia, mes, ano);
23         if (mes > 12) { // reset Jan e Fev
24             mes -= 12;

```

Observe a janela de “**Watches**”. Constate que agora permite observar os valores dos parâmetros da função e da sua variável local `ds`.



Watches

- Local variables
 - `ds = 3`
- Function Arguments
 - `dia = 13`
 - `mes = 14`
 - `ano = 2012`
 - `ano%4 = 0`

Disassembly

Function:

Frame start: 0x7fffffffe510

0x40065e	mov	edx, 0x92492493
0x400663	mov	eax, ecx
0x400665	imul	edx
0x400667	lea	eax, [rdx+rcx*1]
0x40066a	mov	edx, eax
0x40066c	sar	edx, 0x2
0x40066f	mov	eax, ecx



main.c

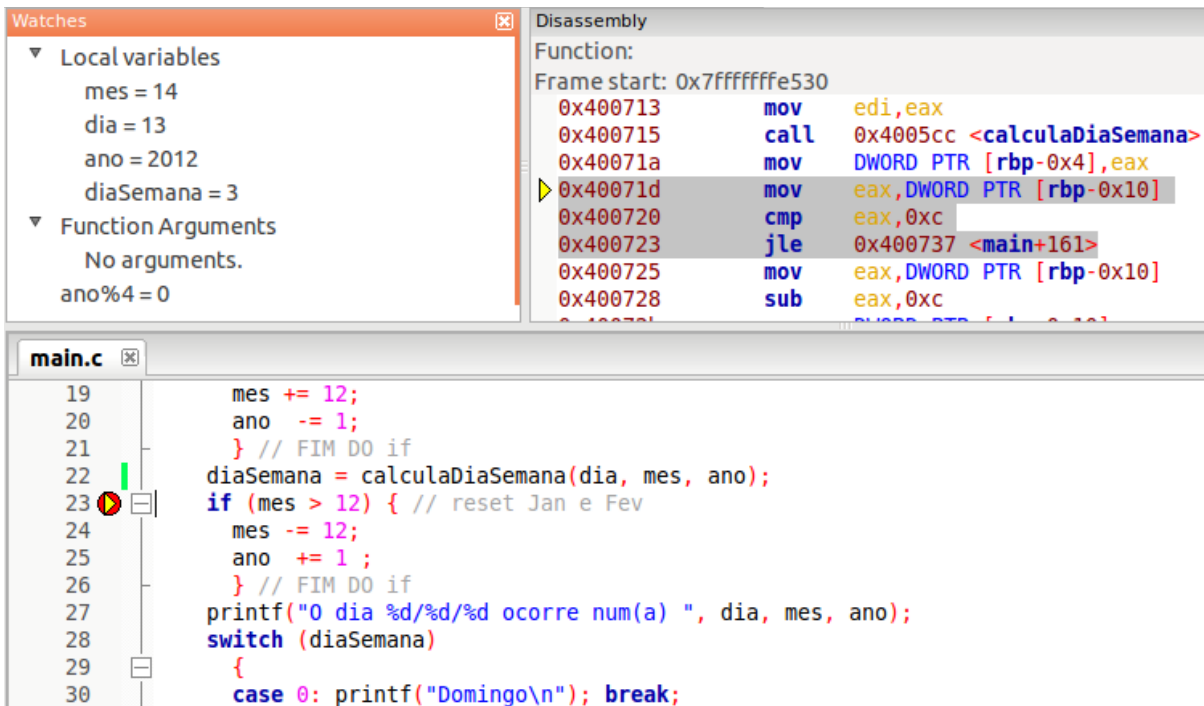
```

1  #include <stdio.h>
2
3  int calculaDiaSemana(int dia, int mes, int ano) {
4      int ds;
5      ds = (dia + 2*mes + 3*(mes+1)/5 + ano + ano/4 - ano/100 + ano/400 + 1) % 7;
6      return ds;
7  }
8
9  int main() // Calcula o dia da semana correspondente a uma data dd/mm/aaaa
10 {
11     int mes, dia, ano, diaSemana;
12     printf("Introduza uma data para a qual deseja saber\n");

```

Para terminar a depuração da função e regressar ao `main()`, podemos clicar no botão “**Step out**”.

Para terminar o tutorial, falta apenas comparar a execução passo-a-passo ao nível do C com a execução passo-a-passo ao nível do *assembly*. Para isso basta ter a janela “**Disassembly**” aberta e com o  visível, de modo a sabermos qual a próxima instrução *assembly* a ser executada. Ao clicar no botão “**Next instruction**” damos ordem para executar a instrução *assembly* apontada. Como exemplo, coloque um ponto de paragem na linha “*if (mes > 12)*” e execute o programa até essa linha. De seguida clique em “**Next instruction**” até o  do *main()* avançar para a linha seguinte. Verificará deste modo que esta instrução em C é implementada com 3 instruções em *assembly* (ver próxima figura).



The screenshot displays a debugger interface with three main panels:

- Watches:** Shows local variables with values: `mes = 14`, `dia = 13`, `ano = 2012`, `diaSemana = 3`. Under "Function Arguments", it shows "No arguments." and `ano%4 = 0`.
- Disassembly:** Shows the assembly code for the current function. The frame start is `0x7fffffff530`. The instructions are:
 - `0x400713 mov edi, eax`
 - `0x400715 call 0x4005cc <calculaDiaSemana>`
 - `0x40071a mov DWORD PTR [rbp-0x4], eax`
 - `0x40071d mov eax, DWORD PTR [rbp-0x10]`** (highlighted with a yellow arrow)
 - `0x400720 cmp eax, 0xc`
 - `0x400723 jle 0x400737 <main+161>`** (highlighted with a yellow arrow)
 - `0x400725 mov eax, DWORD PTR [rbp-0x10]`
 - `0x400728 sub eax, 0xc`
- main.c:** Shows the C source code. A breakpoint is set at line 23:


```

19     mes += 12;
20     ano -= 1;
21 } // FIM DO if
22 diaSemana = calculaDiaSemana(dia, mes, ano);
23 if (mes > 12) { // reset Jan e Fev
24     mes -= 12;
25     ano += 1;
26 } // FIM DO if
27 printf("O dia %d/%d/%d ocorre num(a) ", dia, mes, ano);
28 switch (diaSemana)
29 {
30     case 0: printf("Domingo\n"); break;

```