

Debugger do Code::Blocks

DCC 119 e DCC 120



Debugger (Depurador)



- O *debugger* ou depurador é uma funcionalidade do Code::Blocks e de outras IDEs que permite a verificação passo-a-passo da execução de um algoritmo.
- Impressões, leituras de dados, variáveis, vetores, strings etc. podem ter seus valores verificados a cada execução de uma nova instrução.
- Assim, possíveis falhas no algoritmo podem ser detectadas facilmente.

Debugger (Depurador)

- O *debugger* também serve como uma importante ferramenta para o aprendizado de algoritmos, permitindo que seja feito um “teste de mesa” durante a execução do próprio algoritmo.
- Através do processo de *debugging*, o fluxo de execução de um código pode ser verificado continuamente, por meio da checagem do resultado de cada teste condicional, de cada repetição, de cada operação lógica e matemática etc..

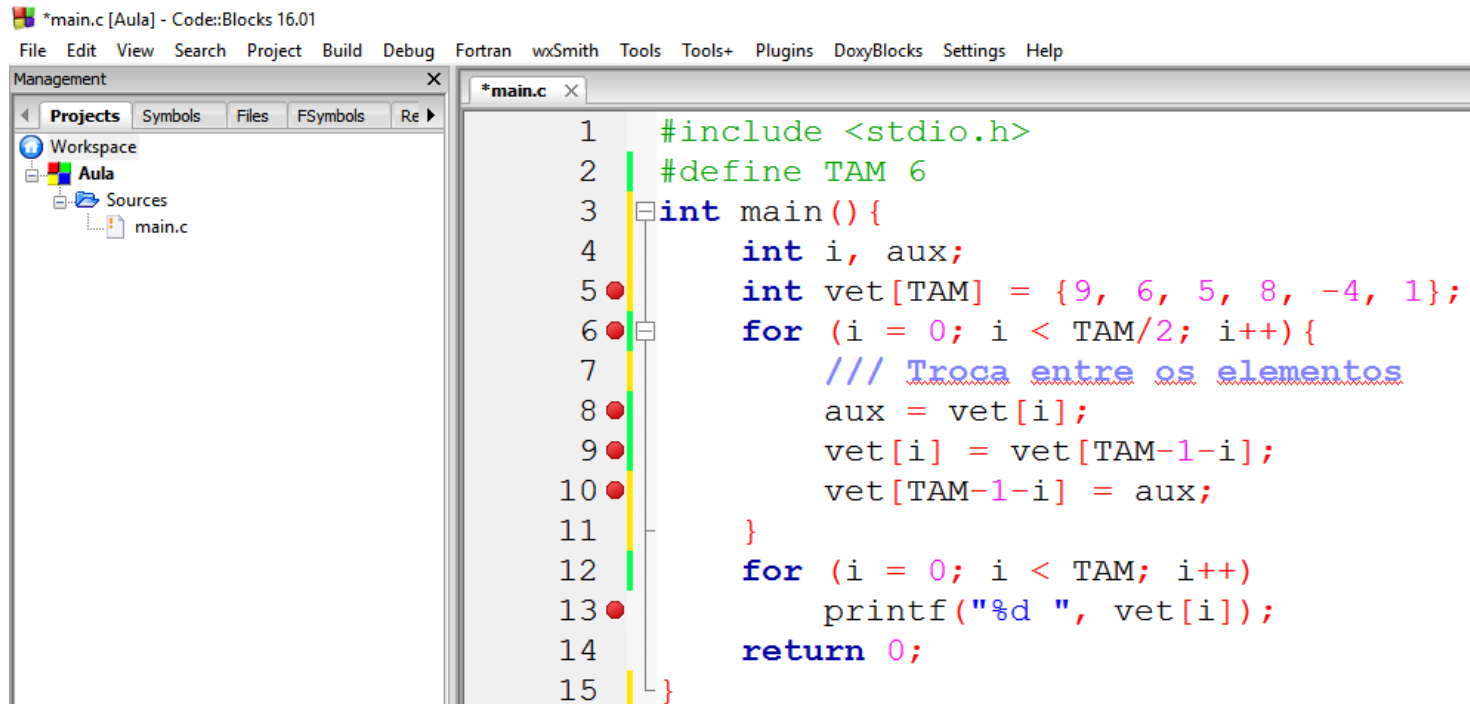
Como utilizar?

- O algoritmo abaixo será utilizado para a explicação do funcionamento do *debugger* do Code::Blocks.
- O algoritmo inverte a ordem dos elementos de um vetor de inteiros.

```
1  #include <stdio.h>
2  #define TAM 6
3  int main() {
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++) {
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
```

Como utilizar?

- Antes de dar início ao *debugging*, é necessário colocar *breakpoints* (pontos de parada) nas linhas cujas execuções devem ser verificadas.
- Basta clicar ao lado direito do número de cada linha desejada. Os hexágonos em vermelho indicam a criação de um *breakpoint*.



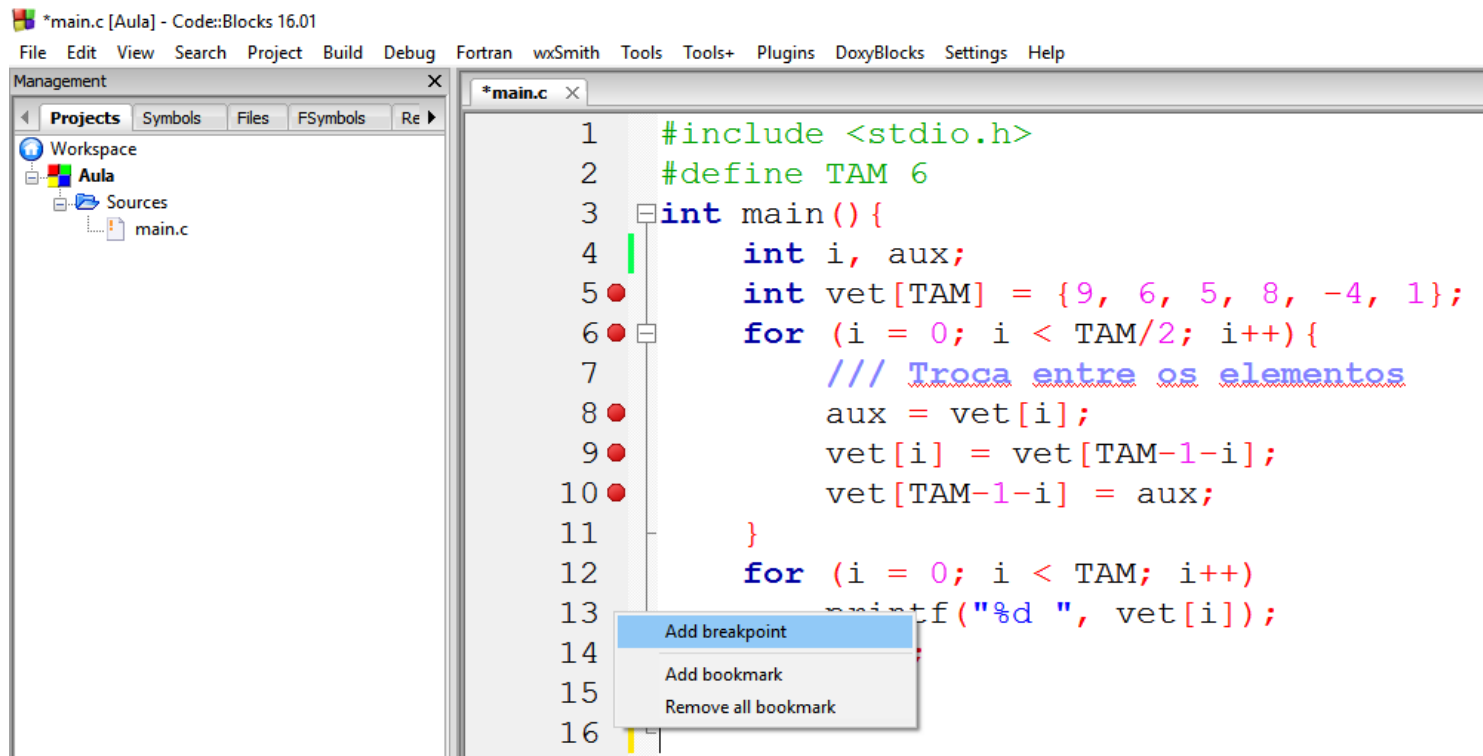
The screenshot shows the Code::Blocks IDE interface. The left sidebar displays the 'Projects' panel with a workspace named 'Aula' containing a source file 'main.c'. The main editor window shows the code of 'main.c' with the following content:

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
```

Red hexagonal breakpoints are placed on the right margin of lines 5, 6, 8, 9, 10, and 13. A yellow vertical bar is visible on the left margin of the editor window.

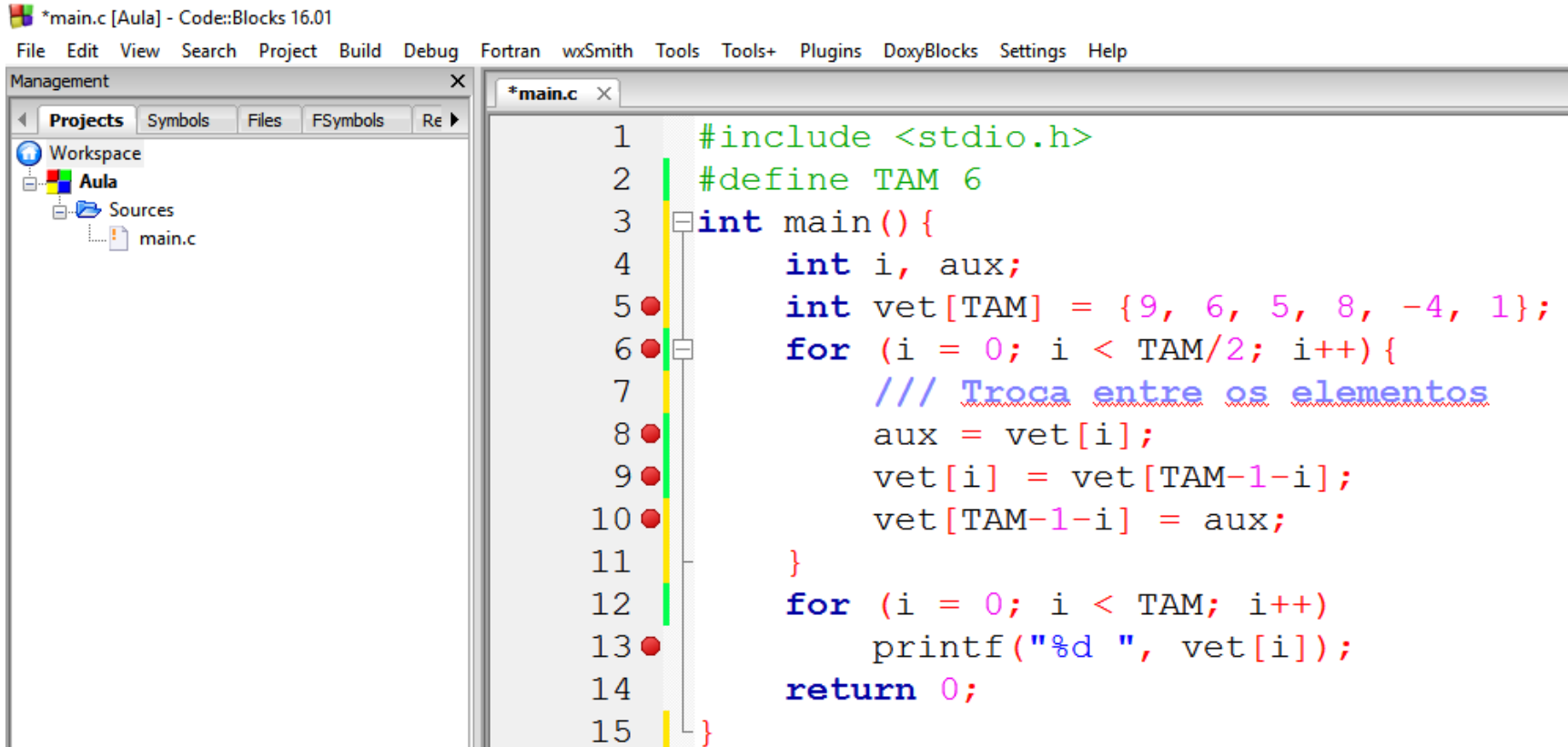
Como utilizar?

- Outras formas de se adicionar um *breakpoint*:
 - Clicar com o botão esquerdo do mouse ao lado direito do número de cada linha desejada e escolher a opção *Add breakpoint*;
 - Colocar o cursor na linha desejada e pressionar a tecla F5.



Como utilizar?

- Com a adição dos *breakpoints*, nas linhas 5, 6, 8, 9, 10 e 13 do algoritmo abaixo, ocorrerá o passo-a-passo da execução.



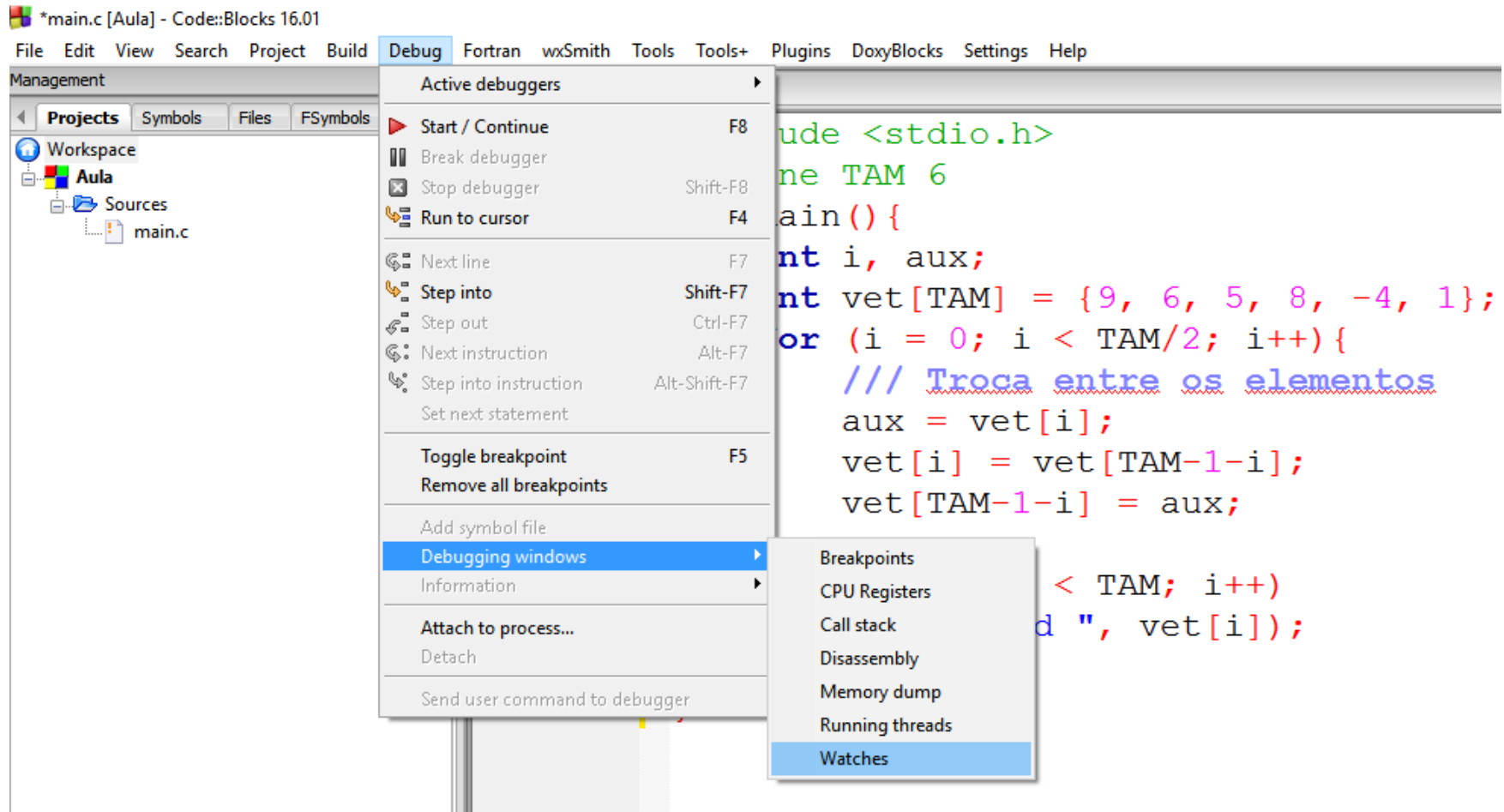
The screenshot shows the Code::Blocks IDE interface. The title bar indicates the file is `*main.c [Aula] - Code::Blocks 16.01`. The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. The left sidebar shows the 'Management' pane with tabs for Projects, Symbols, Files, FSymbols, and Re. The 'Workspace' tree shows a project named 'Aula' containing a 'Sources' folder with the file 'main.c'. The main editor window displays the code for `*main.c` with line numbers 1 through 15. The code is as follows:

```
1  #include <stdio.h>
2  #define TAM 6
3  int main() {
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++) {
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
```

Breakpoints are indicated by red dots on the left margin of the code editor at lines 5, 6, 8, 9, 10, and 13. A yellow vertical bar is positioned to the left of the code, and a green vertical bar is positioned to the right of the code.

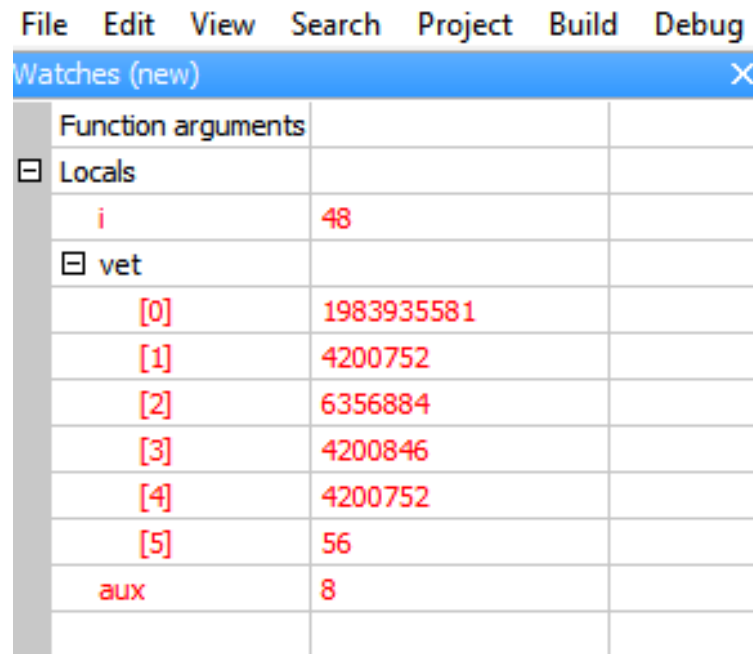
Como utilizar?

- No menu *Debug*, acesse *Debugging windows* e clique em *Watches*.



Como utilizar?

- A janela *Watches* será acionada. Nela é possível acompanhar o conteúdo das variáveis, vetores, strings etc., a cada passo da execução do algoritmo.

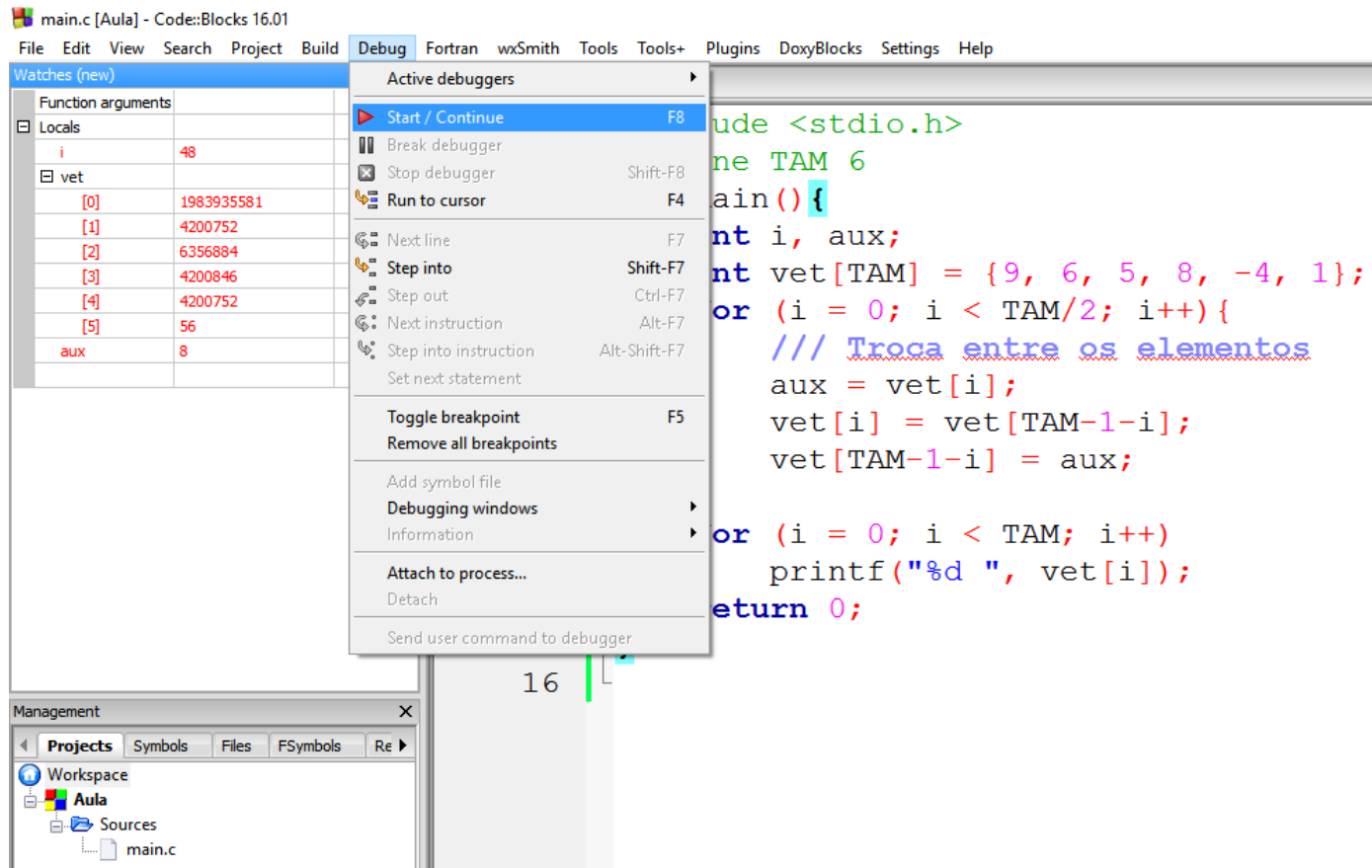


Watches (new) X		
Function arguments		
Locals		
i	48	
vet		
[0]	1983935581	
[1]	4200752	
[2]	6356884	
[3]	4200846	
[4]	4200752	
[5]	56	
aux	8	

Inicialmente, as variáveis `i` e `aux` e o vetor `vet` contém lixo de memória.

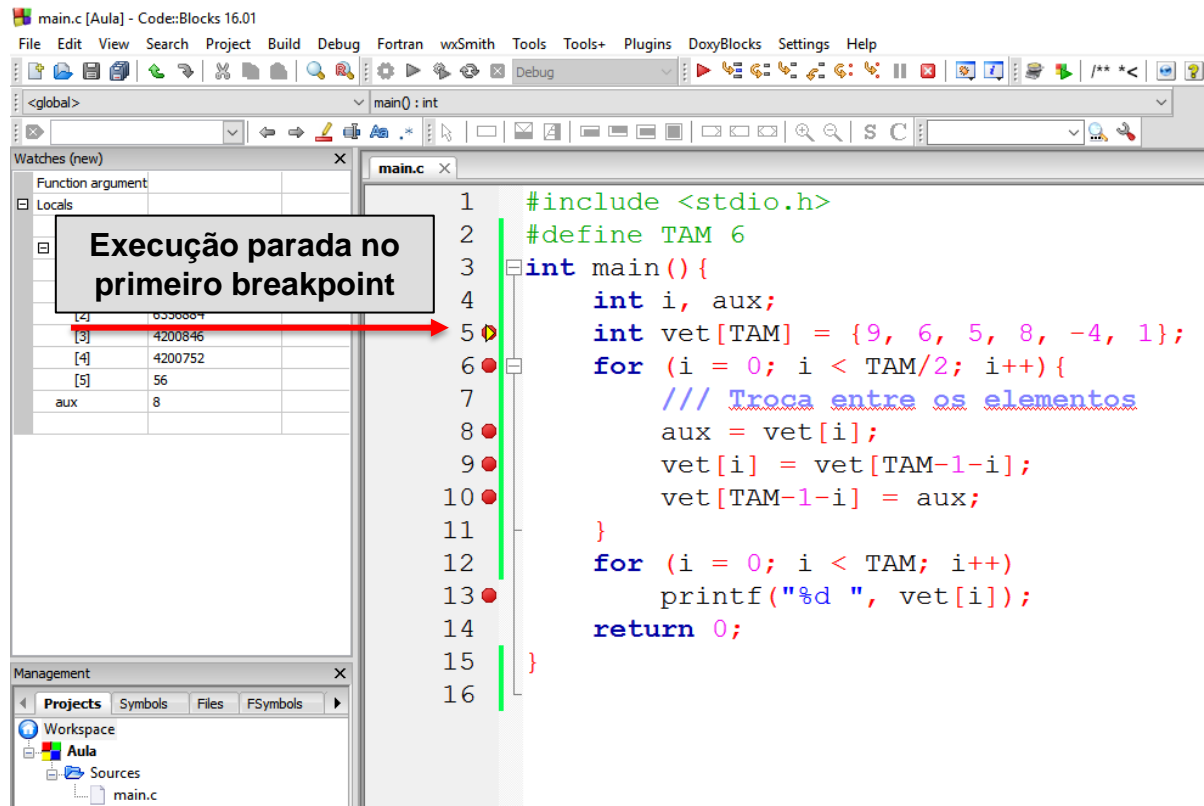
Como utilizar?

- Para dar início ao *debugging*, acesse o menu *Debug* e, em seguida, escolha a opção *Start/Continue* ou pressione a tecla *F8*.



Como utilizar?

- O algoritmo será executado normalmente até que seja encontrado um *breakpoint*. A seta em amarelo indica a linha que será executada (linha 5).



The screenshot shows the Code::Blocks IDE with a C program named `main.c`. The program is in the `main0: int` scope. The code is as follows:

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

A yellow arrow points to line 5, indicating the current execution point. A text box with the text "Execução parada no primeiro breakpoint" is overlaid on the image, pointing to the yellow arrow.

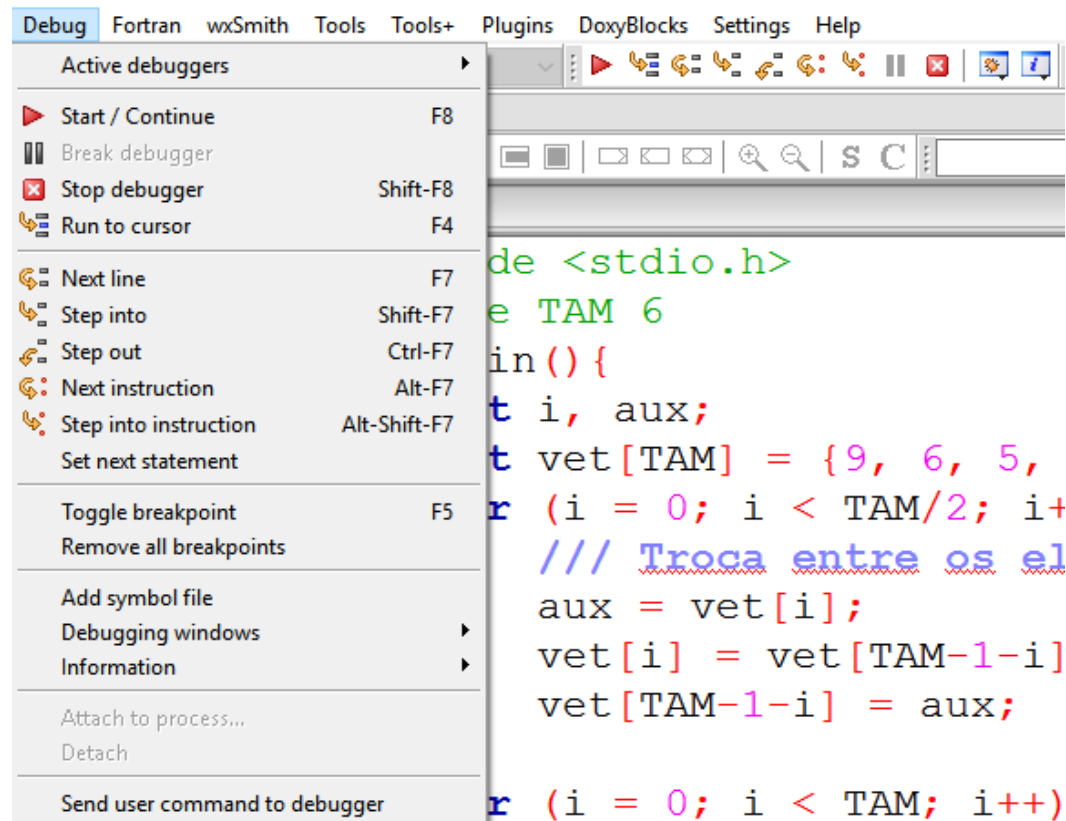
The `Watches (new)` panel on the left shows the following variables and their values:

Function argument	
[2]	6336884
[3]	4200846
[4]	4200752
[5]	56
aux	8

The `Management` panel at the bottom shows the `Projects` tab with the `Aula` project selected, and the `Sources` tab showing the `main.c` file.


Controle da execução

- O controle do fluxo de execução pode ser feito através de alguns comandos dispostos no menu *Debug*, destacado abaixo, que é exibido a partir do início do *debugging*.





Atalhos do menu

Controle da execução

 **Next instruction (Alt-F7):** a próxima instrução é executada.

 **Next line (F7):** a execução é desviada para a próxima linha.


 **Run to cursor (F4):** a execução é desviada para a linha na qual se encontra o cursor.

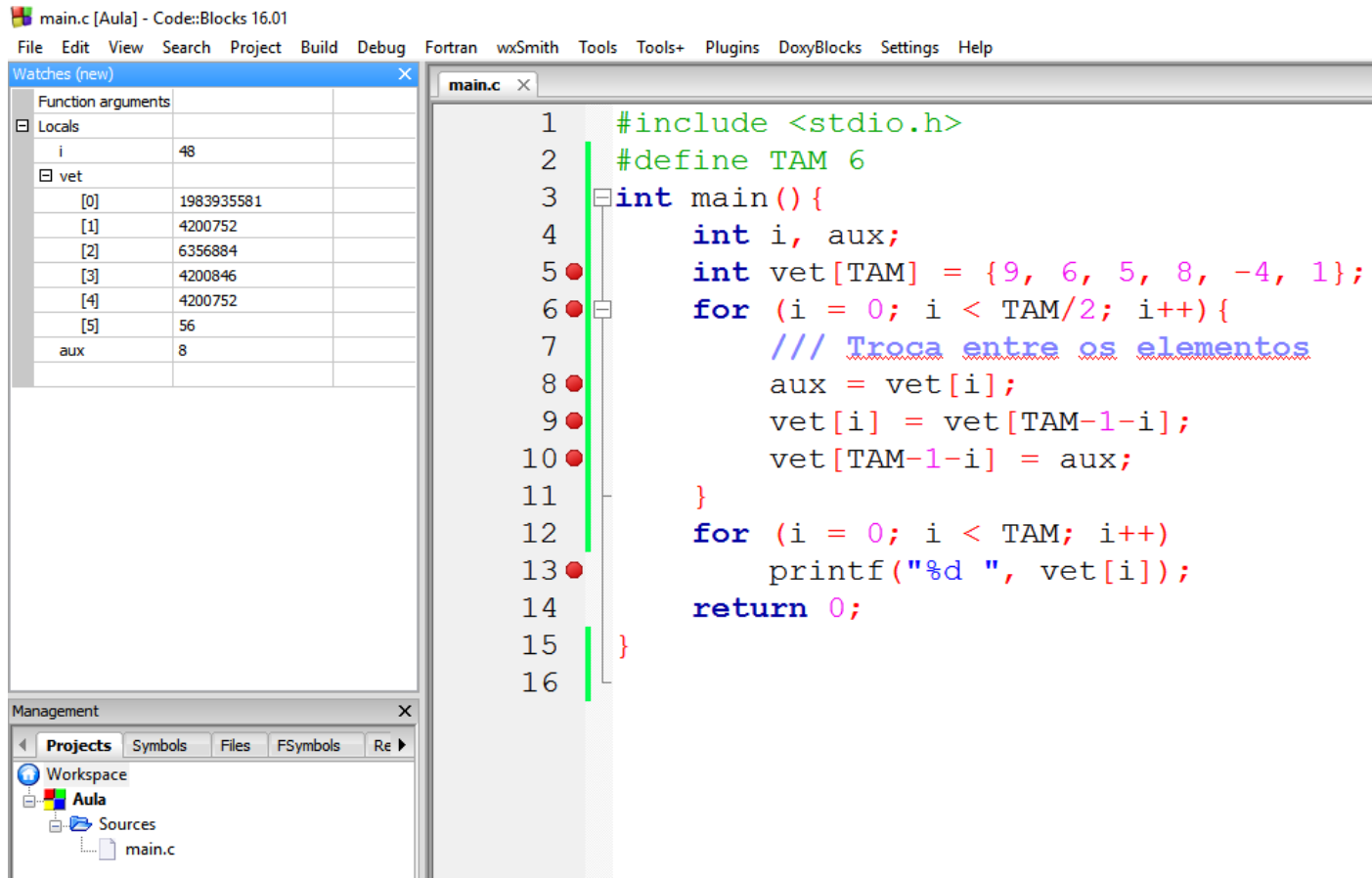
 **Start/Continue (F8):** o processo de *debugging* é iniciado (*start*). O *debugging* é executado passo-a-passo (*continue*).

🖱️ **Step into (Shift-F7):** se a instrução a ser executada for uma chamada de função, a execução é desviada para a função.

🖱️ **Step out (Ctrl-F7):** a função em execução é interrompida. O fluxo retorna para a instrução seguinte à chamada da função.

❌ **Stop debugger (Shift-F8):** O processo de *debugging* é finalizado.

- A seguir, o algoritmo descrito anteriormente será executado passo-a-passo por meio do comando *Next line* (F7)  .



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Watches (new)

Function arguments	
Locals	
i	48
vet	
[0]	1983935581
[1]	4200752
[2]	6356884
[3]	4200846
[4]	4200752
[5]	56
aux	8

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

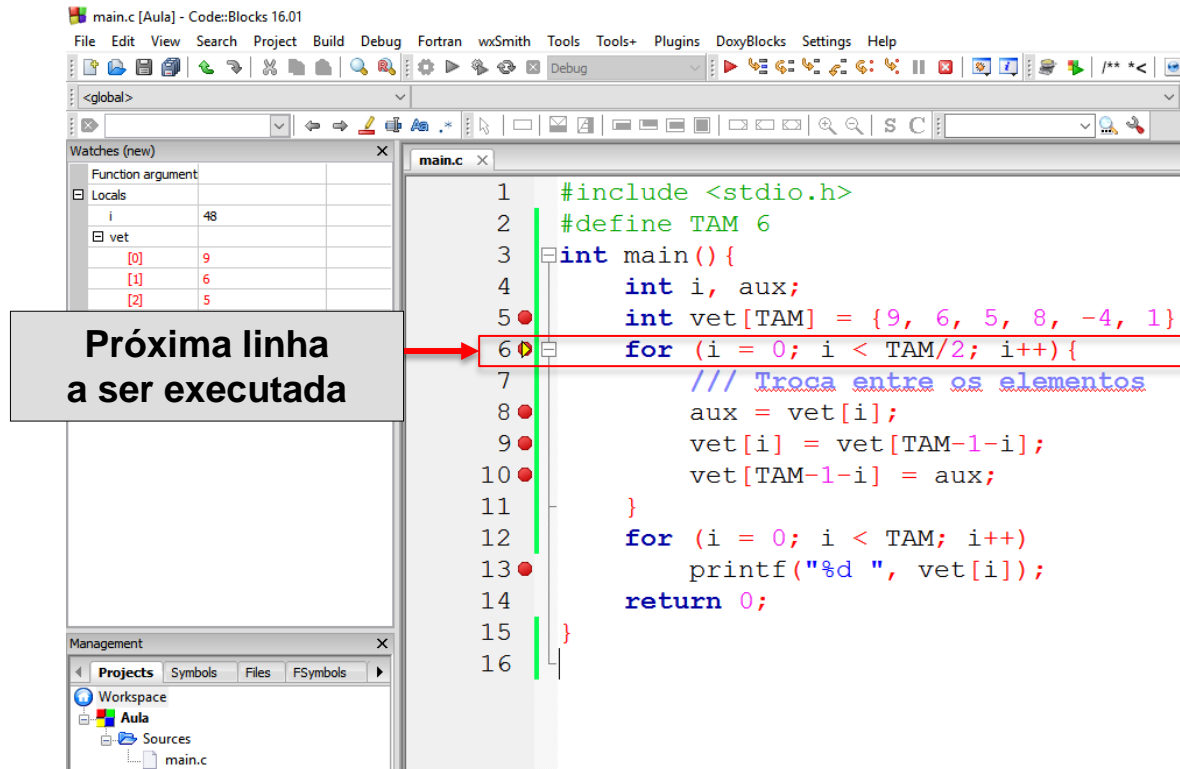
Management

Projects Symbols Files FSymbols Re

Workspace

- Aula
 - Sources
 - main.c

- As linhas em destaque indicam as instruções que serão executadas na sequência.
- Assim, em *Watches* e na tela de impressão os conteúdos exibidos se referem à última execução do algoritmo realizada pelo *debugger*.



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	48
vet	
[0]	9
[1]	6
[2]	5

Próxima linha a ser executada

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

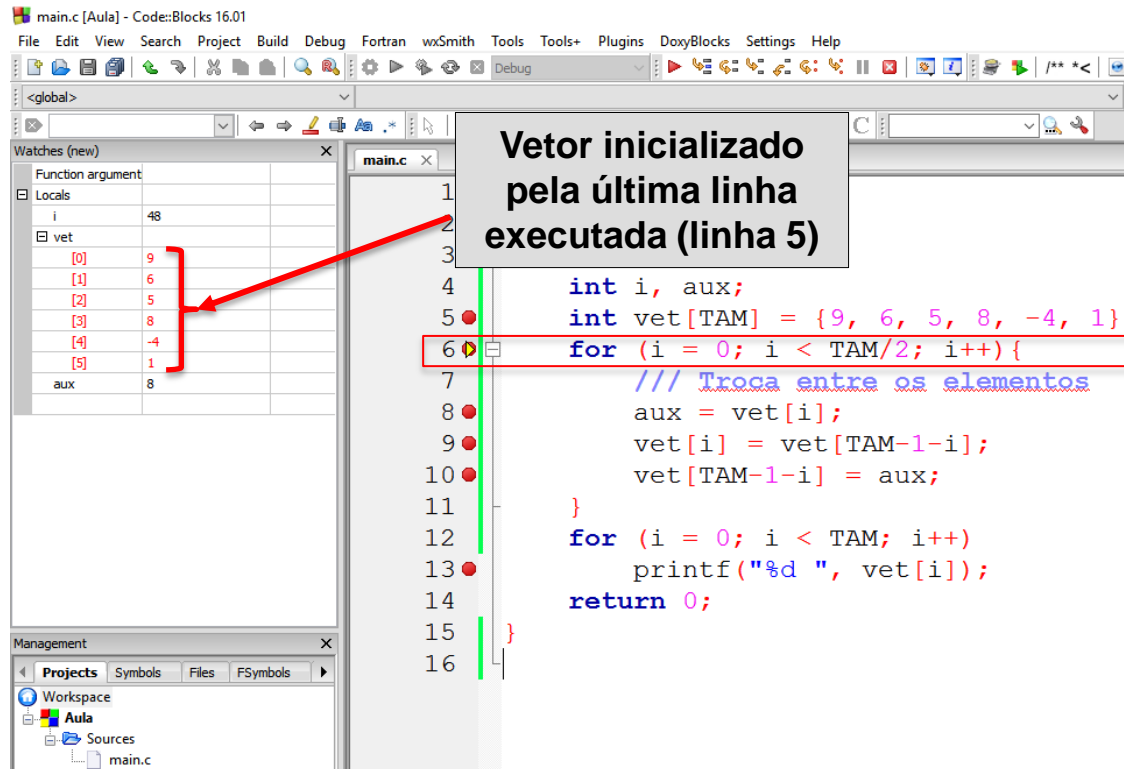
Workspace

Aula

Sources

main.c

- As linhas em destaque indicam as instruções que serão executadas na sequência.
- Assim, em *Watches* e na tela de impressão os conteúdos exibidos se referem à última execução do algoritmo realizada pelo *debugger*.



Vetor inicializado pela última linha executada (linha 5)

```
1  
2  
3  
4 int i, aux;  
5 int vet[TAM] = {9, 6, 5, 8, -4, 1};  
6 for (i = 0; i < TAM/2; i++) {  
7     /// Troca entre os elementos  
8     aux = vet[i];  
9     vet[i] = vet[TAM-1-i];  
10    vet[TAM-1-i] = aux;  
11 }  
12 for (i = 0; i < TAM; i++)  
13     printf("%d ", vet[i]);  
14 return 0;  
15 }  
16
```

Watches (new)	
Function argument	
Locals	
i	48
vet	
[0]	9
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	1
aux	8

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	48
vet	
[0]	1983935581
[1]	42
[2]	63
[3]	4200846
[4]	4200752
[5]	56
aux	8

Início da execução

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

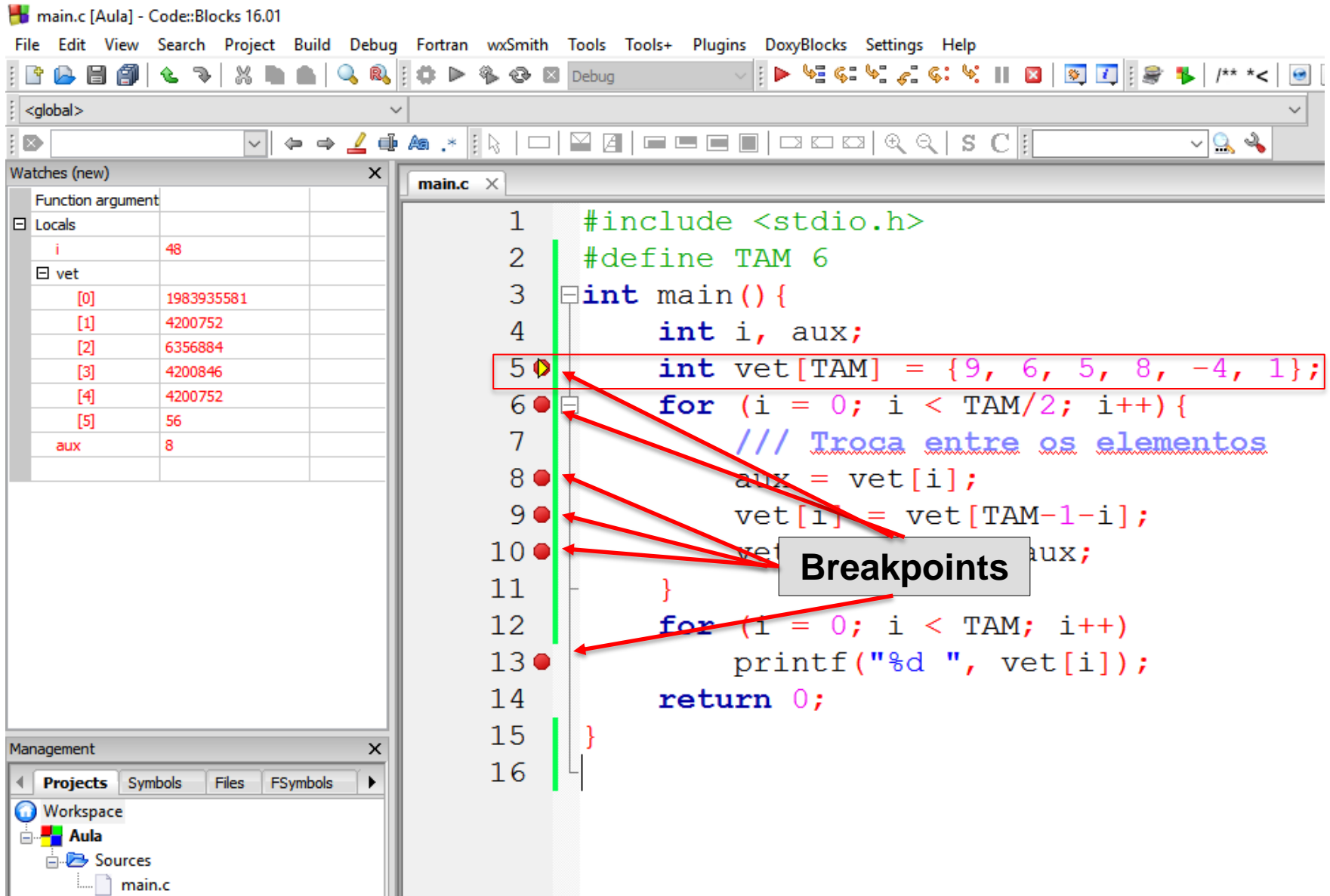
Watches (new)

Function argument	
Locals	
i	48
vet	
[0]	1983935581
[1]	4200752
[2]	6356884
[3]	4200846
[4]	4200752
[5]	56
aux	8

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Breakpoints



Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	48
vet	
[0]	1983935581
[1]	4200752
[2]	6356884
[3]	4200846
[4]	4200752
[5]	56
aux	8

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Variáveis e vetor contendo lixo de memória

```
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	48
vet	
[0]	9
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	1
aux	8

Vetor inicializado

main.c

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	0
vet	
[0]	9
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	1
aux	8

main.c

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Variável inicializada

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	0
vet	
[0]	9
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	1
aux	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3
4  int main(){
5      int i, aux;
6      int vet[TAM] = {9, 6, 5, 8, -4, 1};
7      for (i = 0; i < TAM/2; i++){
8          /// Troca entre os elementos
9          aux = vet[i];
10         vet[i] = vet[TAM-1-i];
11         vet[TAM-1-i] = aux;
12     }
13     for (i = 0; i < TAM; i++)
14         printf("%d ", vet[i]);
15     return 0;
16 }
```

Início da inversão do primeiro elemento com o último

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> main() : int

Watches (new)

Function argument	
Locals	
i	0
vet	
[0]	1
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	1
aux	9

Continuação...

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	0
vet	
[0]	1
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	9
aux	9

Fim da inversão

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	1
vet	
[0]	1
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	9
aux	9

Incremento da variável e teste da condição verdadeiro.

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	1
vet	
[0]	1
[1]	6
[2]	5
[3]	8
[4]	-4
[5]	9
aux	6

main.c

```
1 #include <stdio.h>
2 #define TAM 6
3
4 int main(){
5     int i, aux;
6     int vet[TAM] = {9, 6, 5, 8, -4, 1};
7     for (i = 0; i < TAM/2; i++){
8         /// Troca entre os elementos
9         aux = vet[i];
10        vet[i] = vet[TAM-1-i];
11        vet[TAM-1-i] = aux;
12    }
13    for (i = 0; i < TAM; i++)
14        printf("%d ", vet[i]);
15    return 0;
16 }
```

Início da inversão do segundo elemento com o penúltimo

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	1
vet	
[0]	1
[1]	-4
[2]	5
[3]	8
[4]	-4
[5]	9
aux	6

Continuação...

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	1
vet	
[0]	1
[1]	-4
[2]	5
[3]	8
[4]	6
[5]	9
aux	6

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Fim da inversão

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	2
vet	
[0]	1
[1]	-4
[2]	5
[3]	8
[4]	6
[5]	9
aux	6

Incremento da variável e teste da condição verdadeiro.

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	2
vet	
[0]	1
[1]	-4
[2]	5
[3]	8
[4]	6
[5]	9
aux	5

main.c

```
1 #include <stdio.h>
2 #define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Início da inversão do terceiro elemento com o antepenúltimo

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
i	2

Locals

vet	
[0]	1
[1]	-4
[2]	8
[3]	8
[4]	6
[5]	9
aux	5

Continuação...

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	

Locals

i	2
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

Fim da inversão

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	3
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

Incremento da variável e teste da condição falso.

```
1 include <stdio.h>
2 define TAM 6
3 int main(){
4     int i, aux;
5     int vet[TAM] = {9, 6, 5, 8, -4, 1};
6     for (i = 0; i < TAM/2; i++){
7         /// Troca entre os elementos
8         aux = vet[i];
9         vet[i] = vet[TAM-1-i];
10        vet[TAM-1-i] = aux;
11    }
12    for (i = 0; i < TAM; i++)
13        printf("%d ", vet[i]);
14    return 0;
15 }
16
```

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	0
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

Variável inicializada novamente

Impressões dos elementos do vetor

```
#include <stdio.h>
#define TAM 6
int main(){
    int i, aux;
    int vet[TAM] = {9, 6, 5, 8, -4, 1};
    for (i = 0; i < TAM/2; i++){
        /// Troca entre os elementos
        aux = vet[i];
        vet[i] = vet[TAM-1-i];
        vet[TAM-1-i] = aux;
    }
    for (i = 0; i < TAM; i++){
        printf("%d ", vet[i]);
    }
    return 0;
}
```

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	0
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

Impressões dos elementos do vetor

1

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	1
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	1
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	2
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	2
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9
aux	5

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	3
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	3
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8 5

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	4
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8 5

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	4
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8 5 6

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	5
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8 5 6

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução



main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> main() : int

Watches (new)

Function argument	
Locals	
i	5
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8 5 6 9

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	6
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

Incremento da variável e teste da condição falso.

```
1 include <stdio.h>
2 define TAM 6
3
4 int main(){
5     int i, aux;
6     int vet[TAM] = {9, 6, 5, 8, -4, 1};
7     for (i = 0; i < TAM/2; i++){
8         /// Troca entre os elementos
9         aux = vet[i];
10        vet[i] = vet[TAM-1-i];
11        vet[TAM-1-i] = aux;
12    }
13    for (i = 0; i < TAM; i++)
14        printf("%d ", vet[i]);
15    return 0;
16 }
```

1 -4 8 5 6 9

Execução

main.c [Aula] - Code::Blocks 16.01

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

Watches (new)

Function argument	
Locals	
i	6
aux	5
vet	
[0]	1
[1]	-4
[2]	8
[3]	5
[4]	6
[5]	9

FIM

main.c

```
1  #include <stdio.h>
2  #define TAM 6
3  int main(){
4      int i, aux;
5      int vet[TAM] = {9, 6, 5, 8, -4, 1};
6      for (i = 0; i < TAM/2; i++){
7          /// Troca entre os elementos
8          aux = vet[i];
9          vet[i] = vet[TAM-1-i];
10         vet[TAM-1-i] = aux;
11     }
12     for (i = 0; i < TAM; i++)
13         printf("%d ", vet[i]);
14     return 0;
15 }
16
```

1 -4 8 5 6 9

Management

Projects Symbols Files FSymbols

Workspace

Aula

Sources

main.c

- Para maiores informações sobre o *debugger* do Code::Blocks, acesse o endereço abaixo:

http://wiki.codeblocks.org/index.php/Debugging_with_Code::Blocks