

Esercizio 1 (guidato)

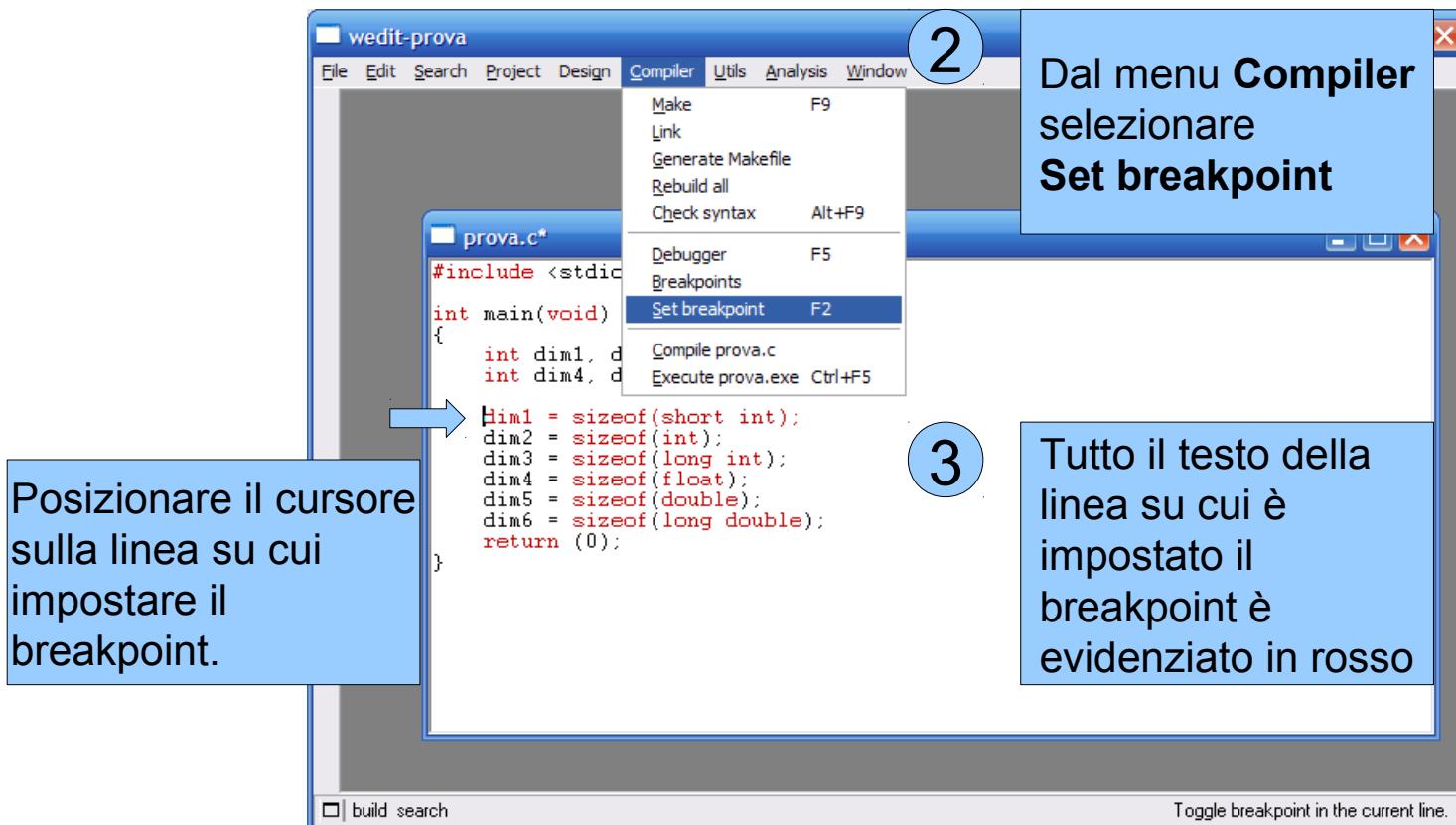
```
#include <stdio.h>
int main(void)
{
    int dim1, dim2, dim3;
    int dim4, dim5, dim6;

    dim1 = sizeof(short int);
    dim2 = sizeof(int);
    dim3 = sizeof(long int);
    dim4 = sizeof(float);
    dim5 = sizeof(double);
    dim6 = sizeof(long double);

    return (0);
}
```

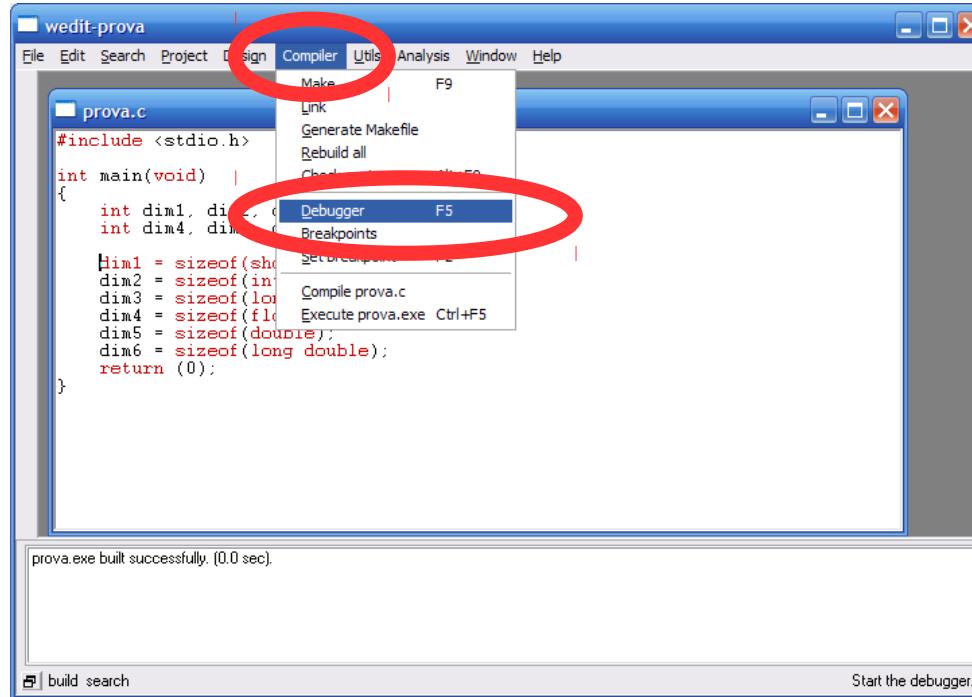
- Copiare, compilare ed eseguire questo programma.
- Utilizzando il debugger si vuole rispondere alle seguenti domande:
 - 1) Quanto vale `dim2` *prima* e *dopo* l'esecuzione del suo assegnamento?
 - 2) Quanti **bit** sono necessari per rappresentare un int?
 - 3) Quanti **bit** sono necessari per rappresentare un float?

Esercizio 1 (guidato)



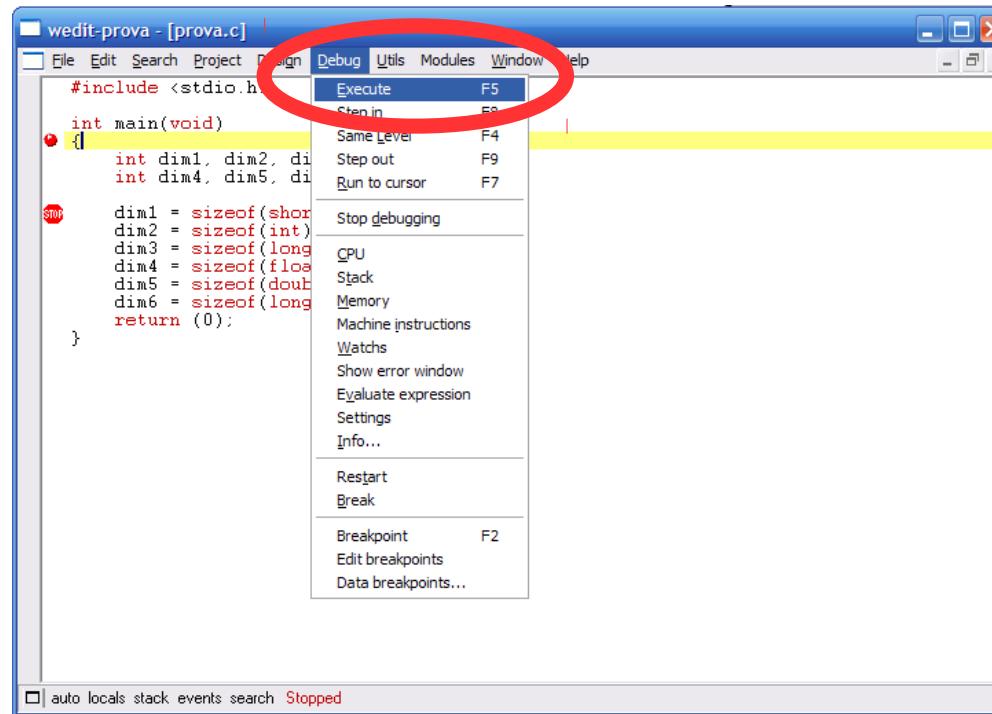
Esercizio 1 (guidato)

Salvare il file (menu File → Save), compilare il programma (menu Compiler → Compile nomefile.c) ed avviare il debugger (menu Compiler → Debugger oppure premere F5)



Esercizio 1 (guidato)

L'esecuzione del programma si blocca sul main. Il pallino rosso indica l'istruzione che si sta per eseguire; il simbolo STOP indica i breakpoint. Per continuare l'esecuzione del programma fino al breakpoint, dal menu Debugger selezionare Execute.



Esercizio 1 (guidato)

L'esecuzione del programma si fermerà al breakpoint. Per controllare il valore delle variabili locali aprire la vista “locals” (evidenziata in figura).

The screenshot shows a Windows-style application window titled "wedit-prova - [prova.c]". The menu bar includes File, Edit, Search, Project, Design, Debug, Utils, Modules, Window, and Help. The main area displays the following C code:

```
#include <stdio.h>
int main(void)
{
    int dim1, dim2, dim3;
    int dim4, dim5, dim6;
    dim1 = sizeof(short int);
    dim2 = sizeof(int);
    dim3 = sizeof(long int);
    dim4 = sizeof(float);
    dim5 = sizeof(double);
    dim6 = sizeof(long double);
    return (0);
}
```

Below the code editor is a "locals" view window containing the following variable values:

dim1	= -370086
dim2	= -370086
dim3	= -370086
dim4	= -370086
dim5	= -370086
dim6	= -370086

The "locals" tab in the bottom navigation bar is circled in red.

Esercizio 1 (guidato)

Per andare avanti una istruzione alla volta selezionare dal menu Debug → Same Level , oppure premere F4

The screenshot shows a debugger interface with the following details:

- Code Editor:** Displays the C code for `prova.c`. A red stop sign icon is at the top left, and a red dot with a yellow border marks a breakpoint on the line `dim3 = sizeof(long);`.
- Watch Window:** Shows variable values:
 - dim1 = 2
 - dim2 = 4
 - dim3 = -370086
 - dim4 = -370086
 - dim5 = -370086
 - dim6 = -370086A red circle highlights the first two entries.
- Debug Menu:** A context menu is open over the breakpoint line, with "Same Level" highlighted (F4).
- Status Bar:** Shows "auto locals stack events search Stopped".

Text Box (Left):

Prima dell'assegnamento
dim2 ha un valore casuale,
dopo vale 4, quindi per
un int su questa macchina
sono richiesti $8 \times 4 = 32$ bit

Esercizio 1 (guidato)

Premere ripetutamente F4 fino a superare l'istruzione relativa a dim4.

The screenshot shows a debugger interface with a blue header bar containing the title "wedit-prova - [prova.c]". The menu bar includes File, Edit, Search, Project, Design, Debug, Utils, Modules, Window, and Help. The main code area contains the following C code:

```
#include <stdio.h>
int main(void)
{
    int dim1, dim2, dim3;
    int dim4, dim5, dim6;

    dim1 = sizeof(short int);
    dim2 = sizeof(int);
    dim3 = sizeof(long int);
    dim4 = sizeof(float);
    dim5 = sizeof(double);
    dim6 = sizeof(long double);
    return (0);
}
```

A red stop sign icon is positioned next to the assignment statement for dim4. A yellow highlight is applied to the line "dim5 = sizeof(double);". The bottom part of the window shows the "locals" tab of the debugger, which displays the following variable values:

dim1 = 2
dim2 = 4
dim3 = 4
dim4 = 4
dim5 = 8
dim6 = -370086

A red circle highlights the value of dim5 in the locals window. The status bar at the bottom indicates "main 15.2" and shows the "locals" tab is selected. A legend at the bottom left of the slide states: "dim4 vale 4, quindi per un float su questa macchina sono richiesti 8 x 4 = 32 bit".

Per far continuare l'esecuzione del programma fino al suo termine dal menu Debug selezionare Execute.

Quanti numeri interi posso rappresentare con una variabile di tipo X?

Supponiamo che uno short int sia codificato con 16 bit (2 byte)...

... 16 bit $\rightarrow 2^{16} \rightarrow$ ho a disposizione 65536 simboli, ma...

... dobbiamo decidere anche se l'intero è *signed* o *unsigned*...

- 1) Caso **short int (signed short int)**: -32768 ... 32767
- 2) Caso **unsigned short int**: 0 ... 65535

Esercizio 2

```
#include <stdio.h>
int main(void)
{
    short int i;
    short int k;

    k = 10000;
    i = 30000 + k;

    return (0);
}
```

1. Copiare, compilare ed eseguire il seguente programma
2. Utilizzando il debug rispondere alle seguenti domande:
 - a) Quanto valgono *i* e *k* prima degli assegnamenti?
 - b) Secondo voi, quanto dovrebbe valere *i* dopo l'assegnamento?
 - c) Quanto vale effettivamente *i* dopo l'assegnamento? Perchè?
3. Modificate il programma, specificando *i* e *k* come variabili *unsigned*... cosa cambia? Il comportamento del programma ora è corretto? Perchè?

È sempre possibile rappresentare qualunque numero intero?

Anche la rappresentazione dei numeri reali soffre di alcuni limiti:

1. Indipendentemente da quanti bit uso per rappresentare un numero reale, tali bit devono essere sempre in numero *finito*...
... se il numero di bit è finito, da qualche parte dovrò approssimare qualcosa...

2. La trasformazione della rappresentazione di un numero reale da una base ad un'altra non è sempre indolare...
...può succedere che, dato un numero reale con un numero di cifre decimali finito in base 10...
... durante la trasformazione di base possa diventare un numero con la parte dopo la virgola addirittura PERIODICA! Quindi, ulteriore approssimazione...

Esercizio 3

```
#include <stdio.h>
int main(void)
{
    float k;
    k = 5.6F;
    k = k - 5.59F;
    return (0);
}
```

1. Copiare, compilare ed eseguire il seguente programma
2. Utilizzando il debug rispondere alle seguenti domande:
 - a) Quanto vale k prima del primo assegnamento?
 - b) Quanto vale k dopo il primo assegnamento?
Quant'e' l'errore di approssimazione?
 - c) Quanto dovrebbe valere, e quanto vale effettivamente k dopo il secondo assegnamento? Perchè?
3. Modificate il programma, specificando k come variabile **double**... cosa cambia?
Quanto vale l'errore di approssimazione?