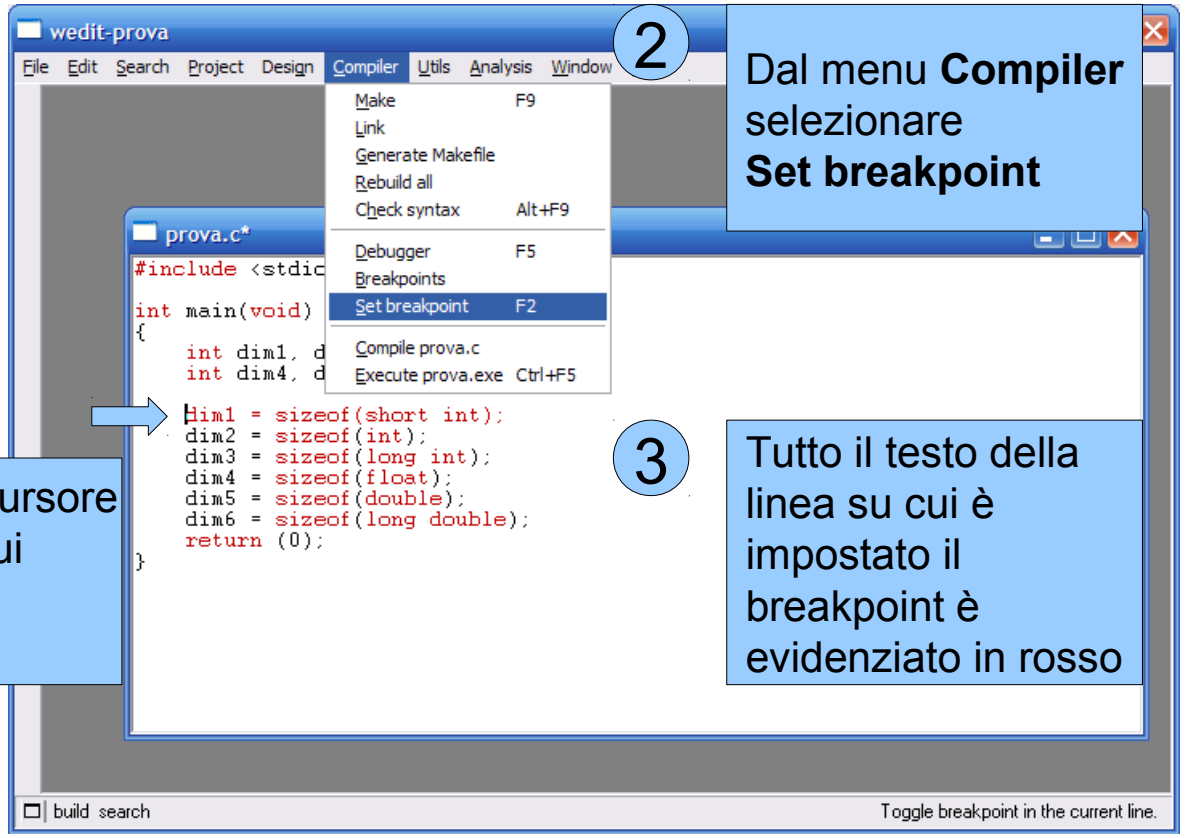
Esercizio 1 (guidato)

```
#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)



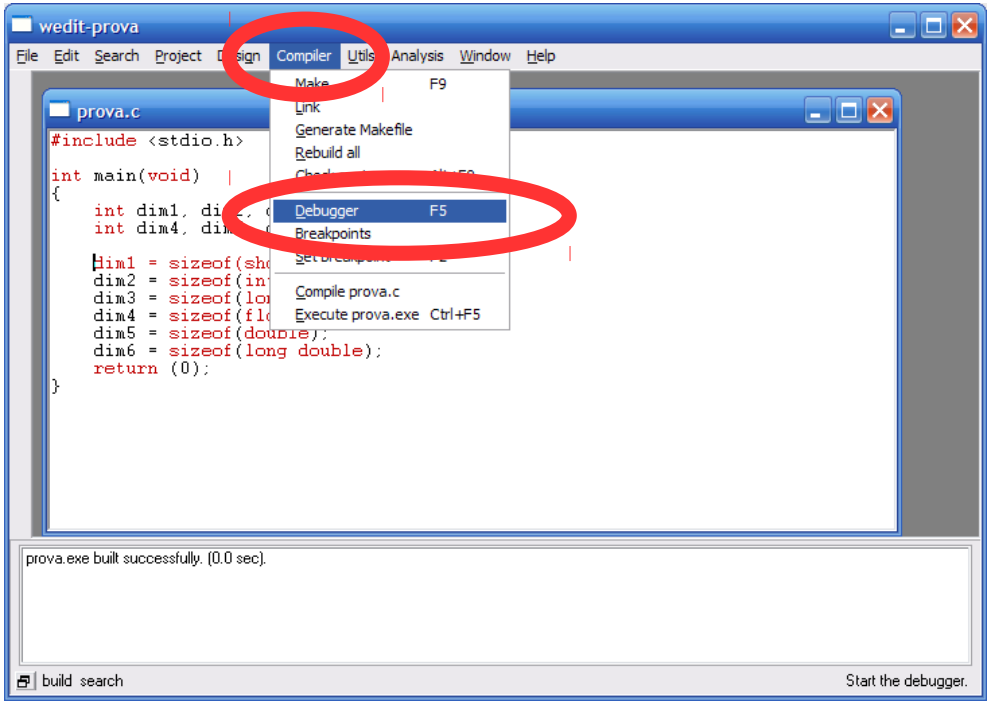
1 Posizionare il cursore sulla linea su cui impostare il breakpoint.

2 Dal menu **Compiler** selezionare **Set breakpoint**

3 Tutto il testo della linea su cui è impostato il breakpoint è evidenziato in rosso

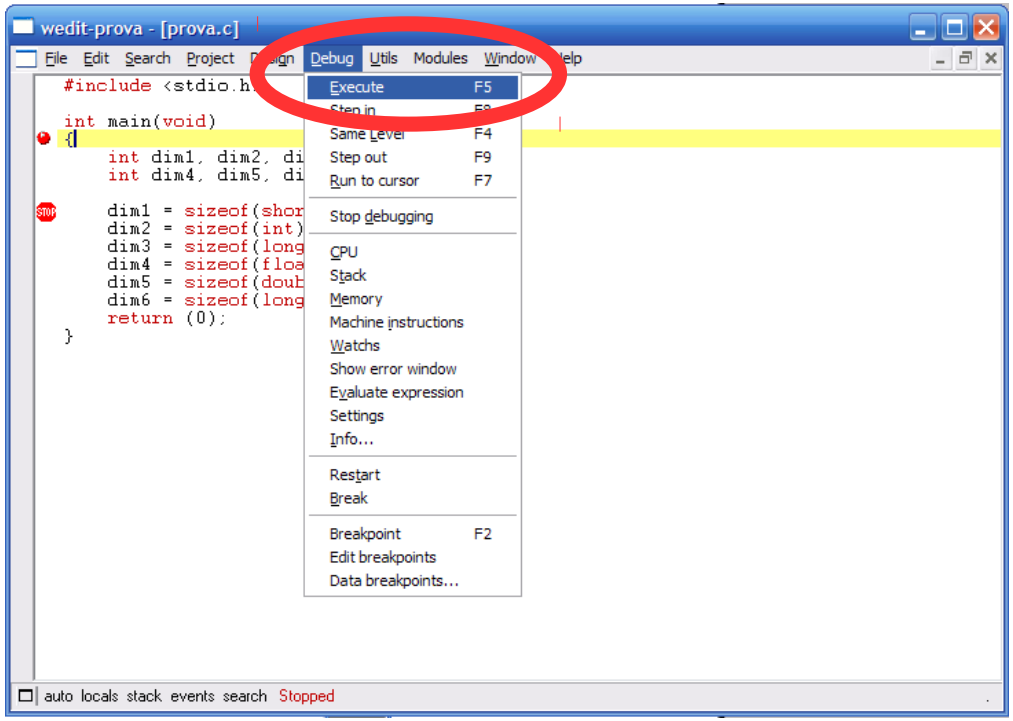
Esercizio 1 (guidato)

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



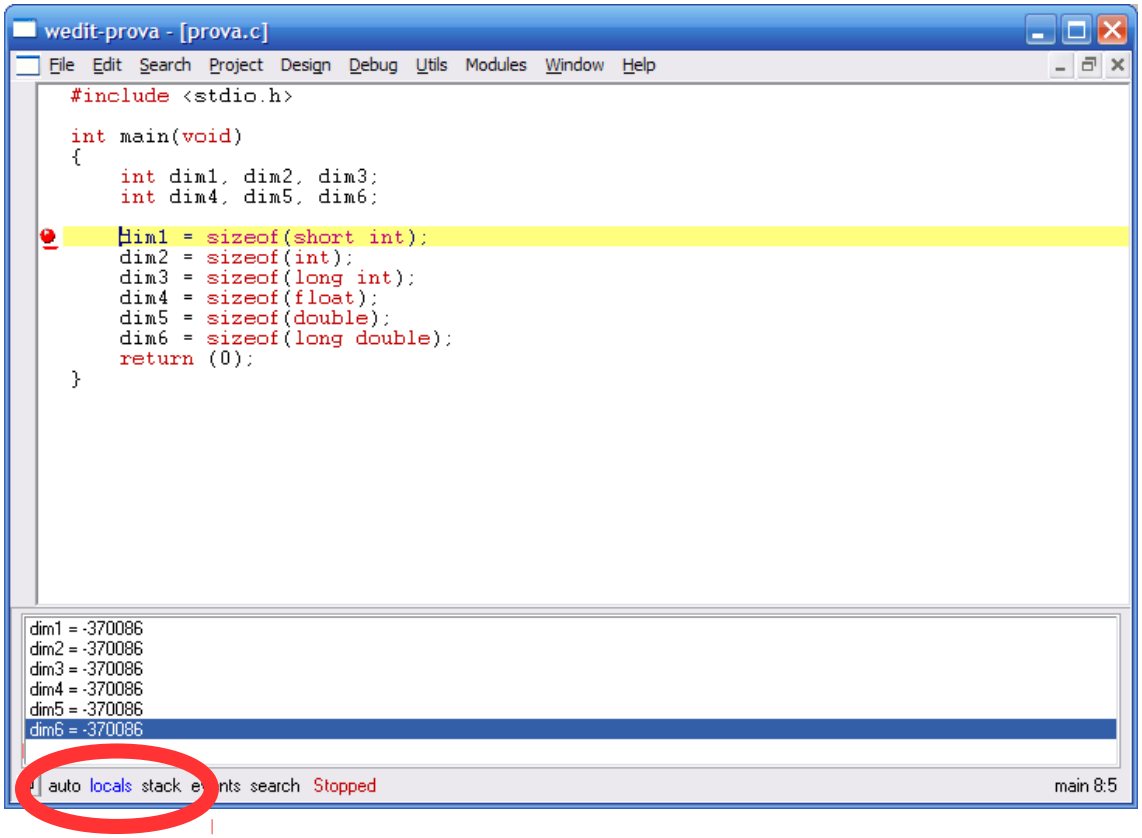
Esercizio 1 (guidato)

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.



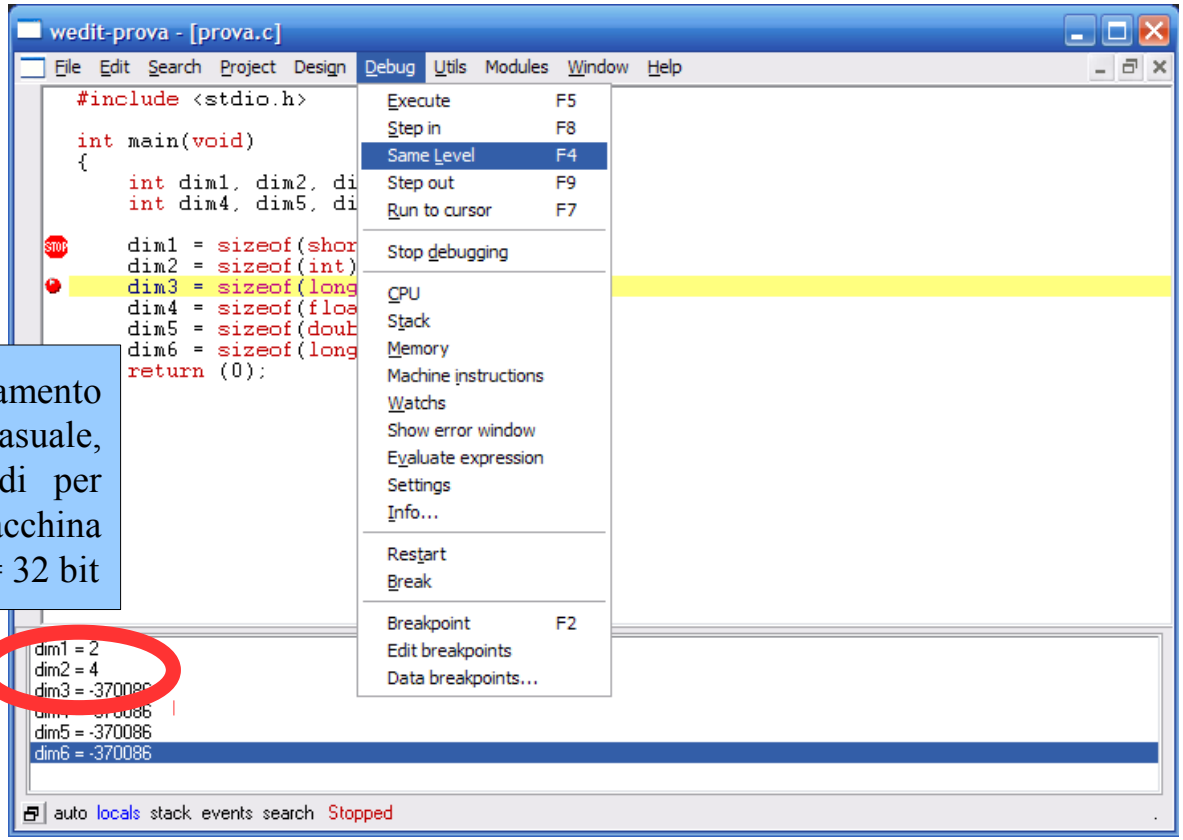
Esercizio 1 (guidato)

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



Esercizio 1 (guidato)

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

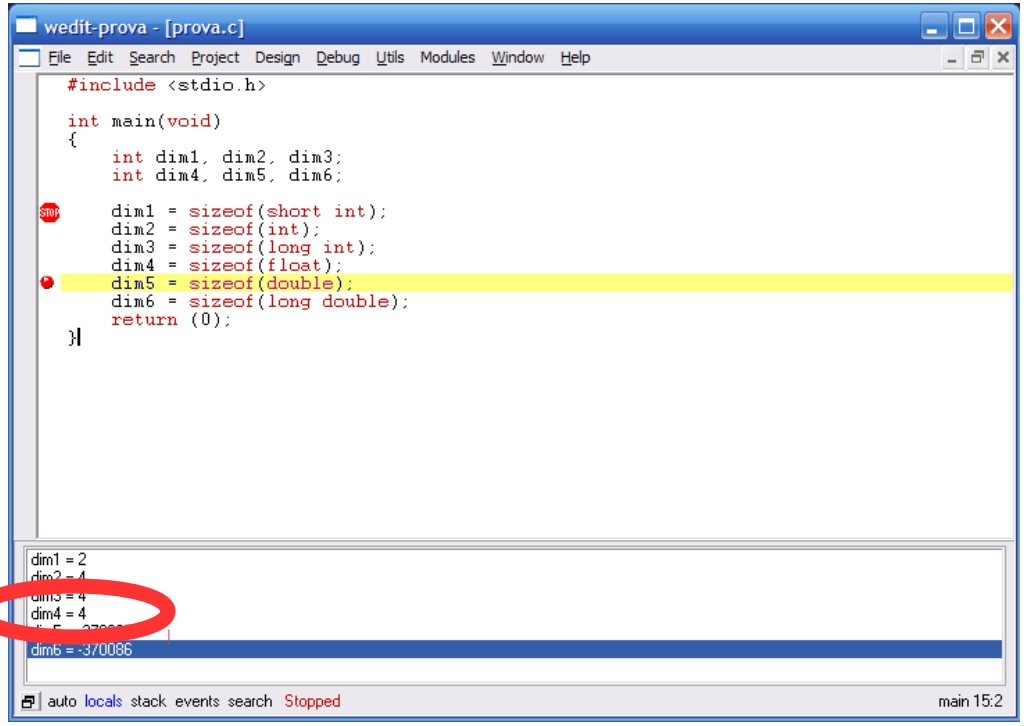


Prima dell'assegnamento `dim2` ha un valore casuale, dopo vale 4, quindi per un int su questa macchina sono richiesti 8 x 4 = 32 bit

Esercizio 1 (guidato)

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

`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 → 2^{16} → 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 qualcosina…

2. La trasformazione della rappresentazione di un numero reale da una base ad un'altra non è sempre indolore…
   …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 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?