Becoming familiar with csd

This section walks through the sample C program **infl**, which is provided on your **csd** distribution disk. It demonstrates **csd**'s basic features, such as how to move the cursor, shift windows, and execute a program under **csd**.

infl is a simple program with a **for** loop. It calculates three different rates of inflation over a span of ten years. The source code is as follows:

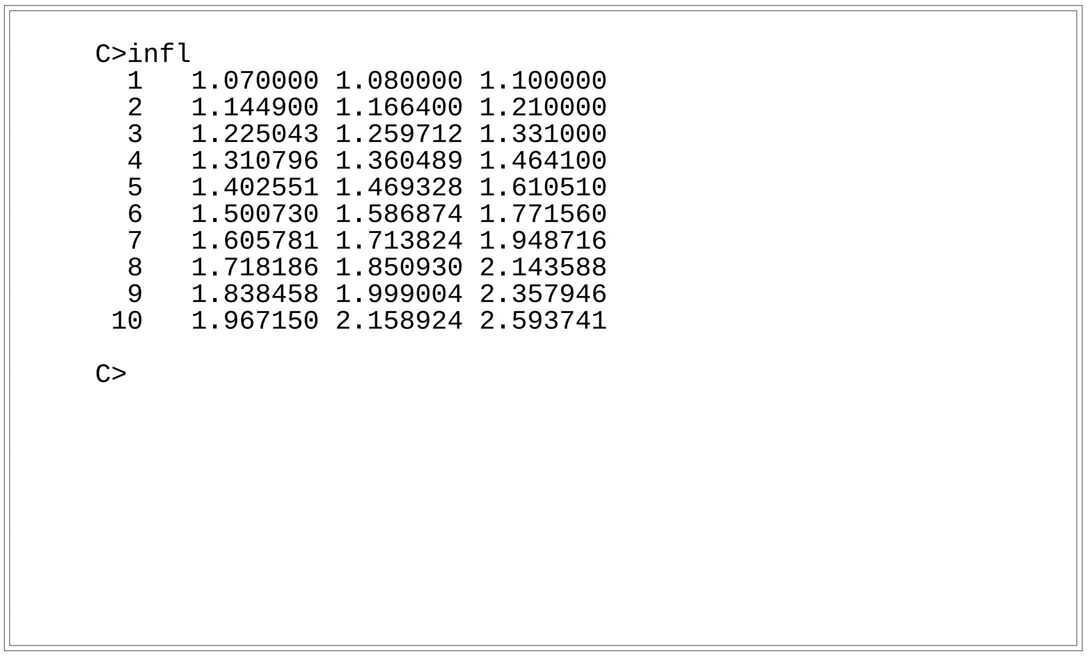
```
#include <stdio.h>
main()
{
        int i;          /* count ten years */
        float w1, w2, w3; /* three inflated quantities */
        char *msg = " %2d\t %f %f %f\n"; /* printf string */
        i = 0;
        w1 = 1.0;
        w2 = 1.0;
        w3 = 1.0;
        for (i = 1; i <= 10; i++) {   /* apply inflation */
                w1 *= 1.07;
                w2 *= 1.08;
                w3 *= 1.10;
                printf (msg, i, w1, w2, w3);
        }
}
```

This program has already been compiled for you. To see its output, type

```
        infl <return>
```

at the MS-DOS prompt. Note that in this tutorial, each line that you type must be followed by **<return>**, unless the text specifically instructs you differently.

Your screen will show the following:



```
C>infl
  1   1.070000 1.080000 1.100000
  2   1.144900 1.166400 1.210000
  3   1.225043 1.259712 1.331000
  4   1.310796 1.360489 1.464100
  5   1.402551 1.469328 1.610510
  6   1.500730 1.586874 1.771560
  7   1.605781 1.713824 1.948716
  8   1.718186 1.850930 2.143588
  9   1.838458 1.999004 2.357946
 10   1.967150 2.158924 2.593741

C>
```

Function keys and what they do

You can manipulate **csd** through the function keys on your keyboard and with the keys on the numeric keypad. These keys let you move the cursor, page through your source code, and move to the beginning or end of your source code.

While using the Mark Williams C compiler, **Let's C**, you have probably become familiar with the editor, MicroEMACS. Switching between MicroEMACS and **csd** while debugging and recompiling doesn't require that you remember two sets of keystrokes. The commands you use to move through text with MicroEMACS will also work with **csd**.

Below is a quick reference list which summarizes all of the command keys and their functions.

FUNCTION	**DESCRIPTION**	**COMMAND**
Find	Find a string of text	<F1> or <esc>1 or <ctrl-S> or <esc>S or <ctrl-R> or <esc>R
Select	Select an option	<F2> or <esc>2
Trace	Trace an expression	<F3> or <esc>3
Run	Run the program	<F4> or <esc>4
Cancel	Cancel the last command	<F5> or <esc>5 or <F5>
Help	Display a help screen	<F6> or <esc>6 or <F6>
Program	Display program window	<F7> or <esc>7
Source	Display source window	<F8> or <esc>8
Evaluation	Enter evaluation window	<F9> or <esc>9
History	Display history window	<F10> or <esc>0
Up	Move cursor up	<↑> or <ctrl-P>
Down	Move cursor down	<↓> or <ctrl-N>
Out	Move to calling function	<←> or <ctrl-B>
In	Undo an **Out**	<→> or <ctrl-F>
Exit	Exit **csd**	<Shift-F1> or <ctrl-X> <ctrl-C>
Current	Return to current line	<Shift-F8> or <ctrl-X>X
Page Up	Move cursor up a page	<ctrl- ↑> or <esc>V
Page Down	Move cursor down a page	<ctrl- ↓> or <ctrl-V>
Delete	Delete a line	<Del> or <ctrl-K>
Insert	Insert a line	<Ins> or <ctrl-O>
Beginning	Beginning of source	<ctrl- ←> or <esc><
End	End of source	<ctrl- →> or <esc>>

See section 5, *Commands reference*, for a complete listing of all the function and keypad keys, the corresponding MicroEMACS command keys, and alternate keystrokes used to control **csd**.

Running csd

Now, try running **infl** under **csd**. If you are using the graphics interface to MWS, invoke **csd** on the program as described in the Introduction.

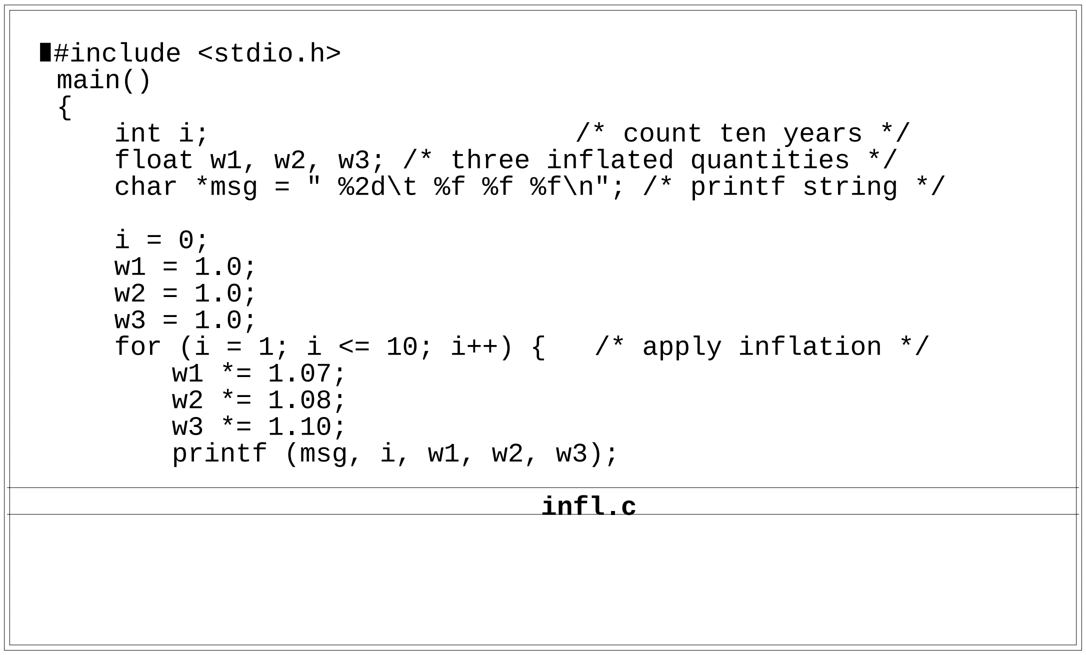
If, however, you are using the **!DOS Escape** option, simply type the following at the prompt:

```
        csd infl
```

In a moment, **csd** will load your program and display the beginning of the source code on your screen.

The screen is divided into two parts. The first and largest of these parts is the *source window*, the top portion of the screen where the source code is displayed. The reverse video line separates the two windows and displays the name of the program module you are currently debugging. Under the reverse video line is the *evaluation window*; it is now empty.

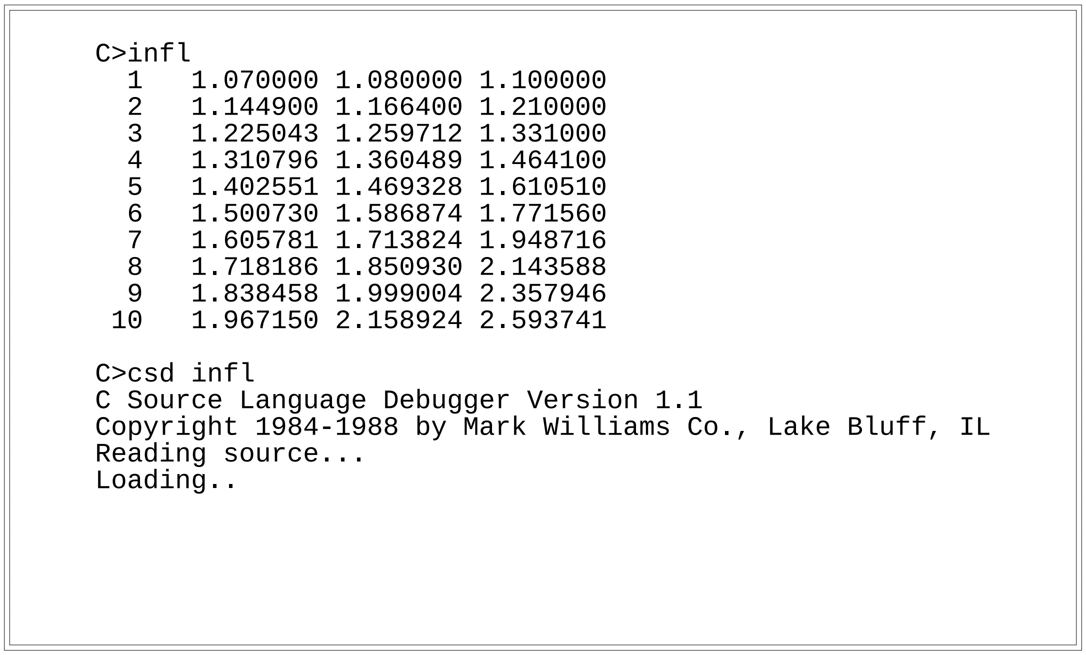
Your screen appears as follows:



```
▮#include <stdio.h>
 main()
 {
     int i;                       /* count ten years */
     float w1, w2, w3; /* three inflated quantities */
     char *msg = " %2d\t %f %f %f\n"; /* printf string */

     i = 0;
     w1 = 1.0;
     w2 = 1.0;
     w3 = 1.0;
     for (i = 1; i <= 10; i++) {   /* apply inflation */
         w1 *= 1.07;
         w2 *= 1.08;
         w3 *= 1.10;
         printf (msg, i, w1, w2, w3);
                                   infl.c
```

The *program window* displays the output of the program you are running. The program window does not appear at the same time as the source and evaluation windows; when you display the program window, the entire screen will be redrawn temporarily.

To see the program window, type the *program* key, **<F7>**. Your screen now shows the following:



```
C>infl
  1   1.070000 1.080000 1.100000
  2   1.144900 1.166400 1.210000
  3   1.225043 1.259712 1.331000
  4   1.310796 1.360489 1.464100
  5   1.402551 1.469328 1.610510
  6   1.500730 1.586874 1.771560
  7   1.605781 1.713824 1.948716
  8   1.718186 1.850930 2.143588
  9   1.838458 1.999004 2.357946
 10   1.967150 2.158924 2.593741

C>csd infl
C Source Language Debugger Version 1.1
Copyright 1984-1988 by Mark Williams Co., Lake Bluff, IL
Reading source...
Loading..
```

To redisplay the source window, press the *source* key, **<F8>.** The source window is now restored.

Moving through the source code

You can move **csd**'s cursor by pressing the cursor movement keys, which are found on the numeric keypad on the right side of your keyboard. If, instead of moving the cursor, numbers appear on the screen when you press any of the keys on the numeric keypad, press the **NumLock** key. This should solve the problem. .PP To move the cursor *down* on the screen, press the <↓> key. Try it. As you can see, the cursor is now positioned at the beginning of the second line of the source code. If you hold the key down, the cursor scrolls down the screen. The MicroEmacs command **<ctrl-N>** also moves the cursor down the screen one line at a time.

To move back *up* the screen, press the <↑> key. As you can see, the cursor has moved to the previous line. Holding this key down scrolls the cursor up the screen. Typing the MicroEMACS command **<ctrl-P>** also moves the cursor up the screen one line at a time.

Note that if you try to move the cursor past the beginning or the end of your source file, **csd** prints the error message **try Help** at the bottom of your screen.

If you wish to move the cursor more than a few lines, it is handier to use the *page* keys <PgUp> and <PgDn> . These keys move through your source program a page at a time. A *page* is one window of lines. Thus, if your source window displays 19 lines of text, as it does when **csd** starts up, then pressing <PgDn> displays the next 19 lines of text (unless, of course, you are at the end of the source file). When you type <PgUp> , the previous 19 lines of text are displayed. If you are within 19 lines of the end of the source file, <PgDn> will move the cursor to the end of your file; likewise, if you are within 19 lines of the beginning of your source file, <PgDn> will move the cursor to the beginning of the file.

The MicroEMACS **<ctrl-V>** command will also move the cursor to the next page of lines; **<esc> V** moves the cursor to the previous page of lines.

To move the cursor to the end of your source file, type <End> . Try it. The cursor is now positioned at the end of the last line of **infl.c**. To return to the beginning of your source code, type <Home> Try it. The cursor is again positioned at the beginning of **infl.c**.

<ctrl-A> and **<ctrl-E>** move the cursor to the beginning and end of the source, respectively.

Finding text

If you want to locate a specific line in your program, you could hunt for it by scrolling through the text line by line or page by page. However, to ease the search, **csd** has a string search feature. With the *find* key, **<F1>**, you can type in a string that **csd** will find for you.

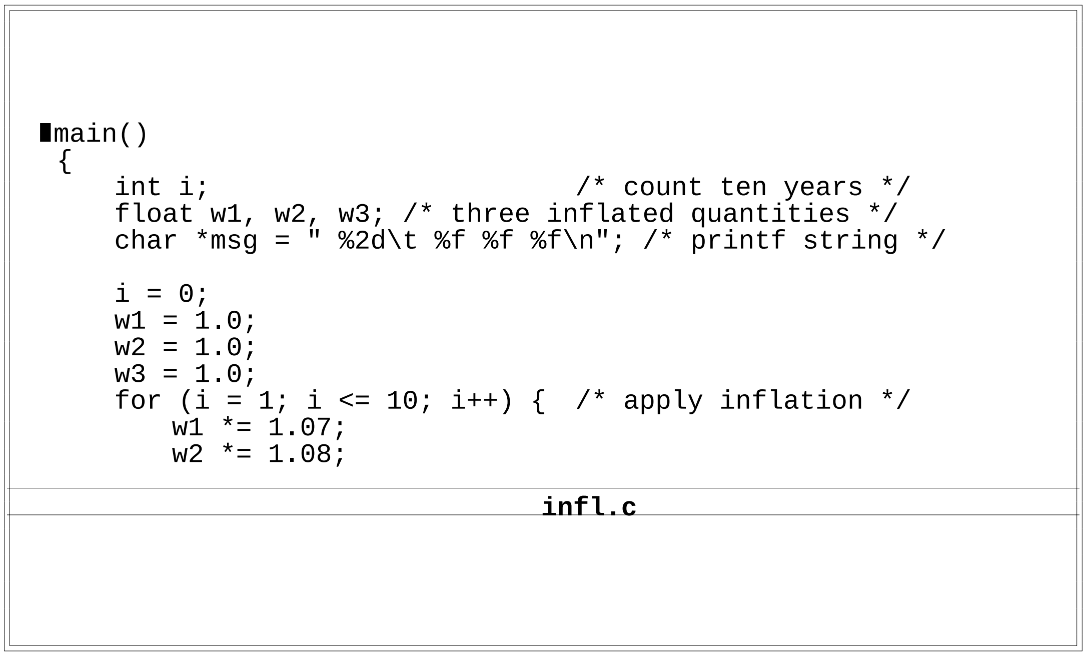
To see how this works, type **<F1>**. At the bottom of the screen, **csd** prints the following prompt:

```
find:[pattern] ↑ ↓ <Home> <End> Cancel<F5> Help<F6>
```

To find the string **main** in **infl.c**, type

```
        main
```

followed by **<return>** or **<↓>**. The screen now displays the following:



```
█main()
 {
    int i;                       /* count ten years */
    float w1, w2, w3; /* three inflated quantities */
    char *msg = " %2d\t %f %f %f\n"; /* printf string */

    i = 0;
    w1 = 1.0;
    w2 = 1.0;
    w3 = 1.0;
    for (i = 1; i <= 10; i++) {  /* apply inflation */
        w1 *= 1.07;
        w2 *= 1.08;
                                   infl.c
```

You can control the direction of search with the <↑> and <↓> keys. The <↑> key searches for the string above the current position of the cursor, and the <↓> key searches below it.

The MicroEMACS commands **<ctrl-S>** and **<ctrl-R>** will also search for a string. You can also use these commands to search for strings in your source code:

<esc>S
<esc>R
<esc>1

Exiting from csd

If you wish to exit from **csd**, press **<Shift-F1>**. **csd** will exit and restore the program window. You can use the **<Shift-F1>** key to exit from **csd** at any time during debugging. However, if you are in the middle of typing something in the evaluation window, you must type **<ctrl-U>** before you exit **csd**. Type **<Shift-F1>**. You must also exit the help screens before trying to exit **csd**.

Setting tracepoints

As noted in section 1, **csd** lets you set *tracepoints* within a program to control its execution. When you set a tracepoint and execute a program, the program stops executing at the tracepoint. You can examine variables and see what has happened to them up to that point in the program. This allows you to run a program step by step, so you discover more easily the point at which your program fails.

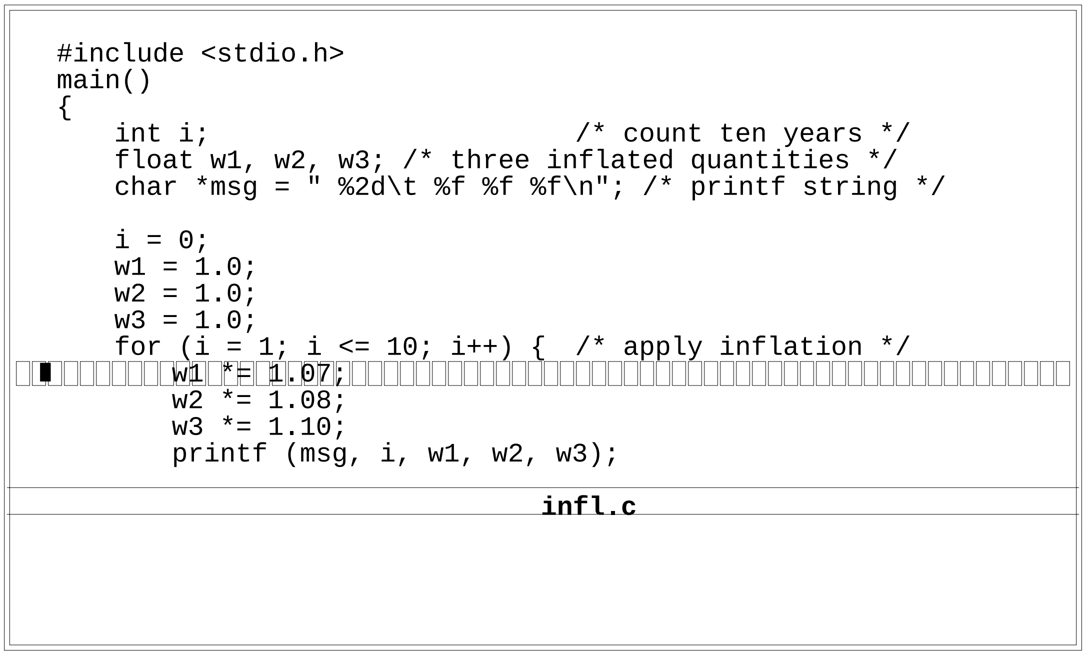
To see how you can set tracepoints, invoke csd on the program **infl** by typing:

```
csd infl
```

Using the arrow keys, or the MicroEMACS cursor control keys, position the cursor at the first executable statement after the **for** statement:

```
w1 *=1.07;
```

Press the *trace* key, **<F3>**; pressing **<F3>** will set a tracepoint on the line where the cursor is positioned. As you can see, setting a tracepoint on a line of code causes that line to be highlighted on your screen. Your screen should now look like this:



```
#include <stdio.h>
main()
{
    int i;                      /* count ten years */
    float w1, w2, w3; /* three inflated quantities */
    char *msg = " %2d\t %f %f %f\n"; /* printf string */

    i = 0;
    w1 = 1.0;
    w2 = 1.0;
    w3 = 1.0;
    for (i = 1; i <= 10; i++) {  /* apply inflation */
        w1 *= 1.07;
        w2 *= 1.08;
        w3 *= 1.10;
        printf (msg, i, w1, w2, w3);
                              infl.c
```

If you were to type **<F3>** again, the tracepoint would be removed and the line would return to its normal intensity. For now, leave the tracepoint on.

Executing to the tracepoint

Now, run the sample program to the traced line. To do so, press **<F4>** to run, and **<F3>** to trace.

This combination of keys tells **csd** to execute your program to the *current statement*. The current statement is the next line of code to be executed. In this example, the program began execution at its beginning, and continued to execute until it encountered the tracepoint you set a few moments ago.

Note that the cursor is again positioned where you set the tracepoint.

Again, type **<F4>**and **<F3>. csd** runs until it again encounters the traced line. While it is executing the program to the tracepoint, it will temporarily display the program window. When **csd** has executed to the tracepoint, it restores the source window.

To see the program's output, press **<F7>.** The screen will show the following:

```
    4 1.310796 1.360489 1.464100
  5   1.402551 1.469328 1.610510
  6   1.500730 1.586874 1.771560
  7   1.605781 1.713824 1.948716
  8   1.718186 1.850930 2.143588
  9   1.838458 1.999004 2.357946
 10   1.967150 2.158924 2.593741

C> csd infl
C Source Language Debugger Version 1.1
Copyright 1984-1988 by Mark Williams Co., Lake Bluff, IL
Reading source..
Loading...

C> csd infl
C Source Language Debugger Version 1.1
Copyright 1984-1988 by Mark Williams Co., Lake Bluff, IL
Reading source...
Loading...
 1    1.070000 1.080000 1.100000
```

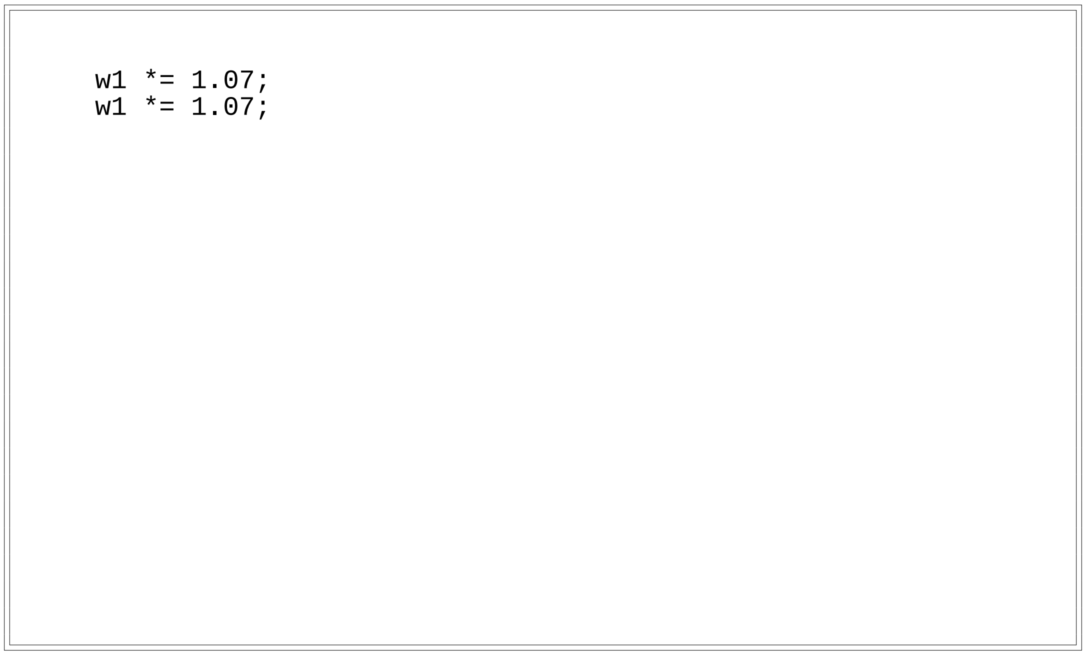
Because the tracepoint is set within the **for** loop, you can see what the program generates with each iteration.

Return to the source window by typing **<F8>**. Press the trace key **<F3>**; this removes the tracepoint. The line returns to normal intensity, and the cursor remains at that line.

History window

Every time your program stops at a tracepoint, **csd** writes a copy of the traced line into the *history window*. This window is where **csd** logs all the statements that have been traced during the execution of your program.

To see the history window, type the *history* key, **<F10>.** If you have followed the steps of the tutorial so far, your screen will look like this:



```
    w1 *= 1.07;
    w1 *= 1.07;
```

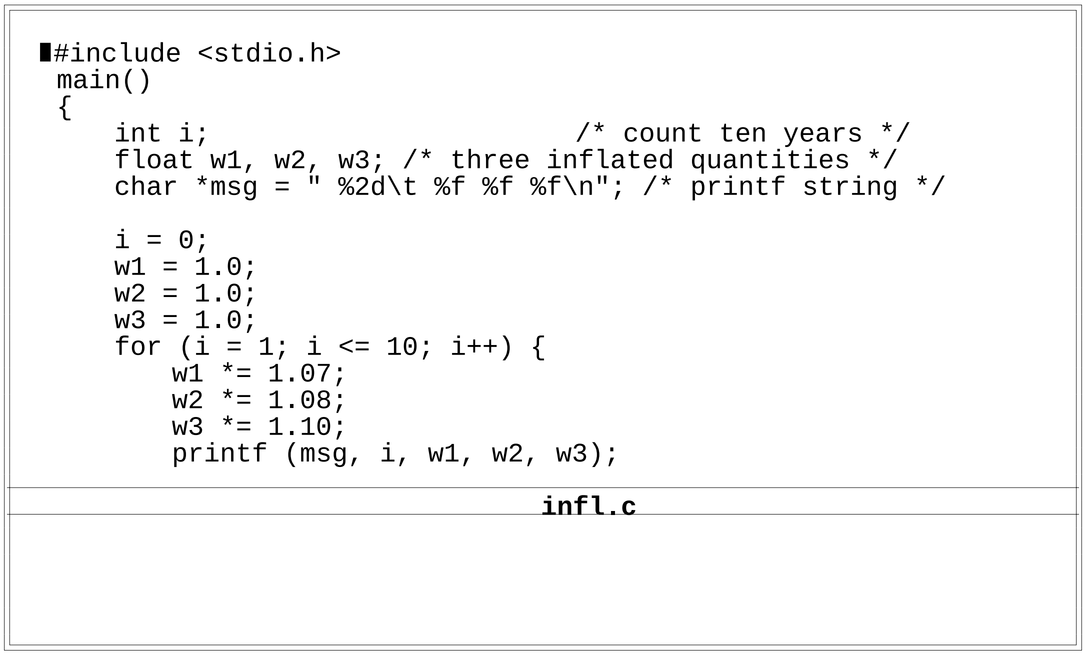
The same statement appears twice because the program stopped executing there twice.

Type **<F8>** to return to the source window.

Single-stepping through a program

With **csd**, you can step through a program a line at a time and examine its execution in detail. This procedure is called *single-stepping.*

To see how single-stepping works, you should first restart your program. You need to do this because if you have been following the steps of this tutorial, the program is midway through its execution. To restart a program, type **<F4>** followed by the *begin* key, <Home> . **csd** reloads the program, then returns the cursor to the beginning of the source file. Your screen appears as follows:



```
▮#include <stdio.h>
 main()
 {
     int i;                       /* count ten years */
     float w1, w2, w3; /* three inflated quantities */
     char *msg = " %2d\t %f %f %f\n"; /* printf string */

     i = 0;
     w1 = 1.0;
     w2 = 1.0;
     w3 = 1.0;
     for (i = 1; i <= 10; i++) {
         w1 *= 1.07;
         w2 *= 1.08;
         w3 *= 1.10;
         printf (msg, i, w1, w2, w3);
                                   infl.c
```

Now, to execute your program one step at a time, press **<F4>** followed by **<return>**. The program window (which shows the program's output) is displayed briefly; then, the source window is restored, with the cursor positioned at the first executable statement in the program:

```
i = 0;
```

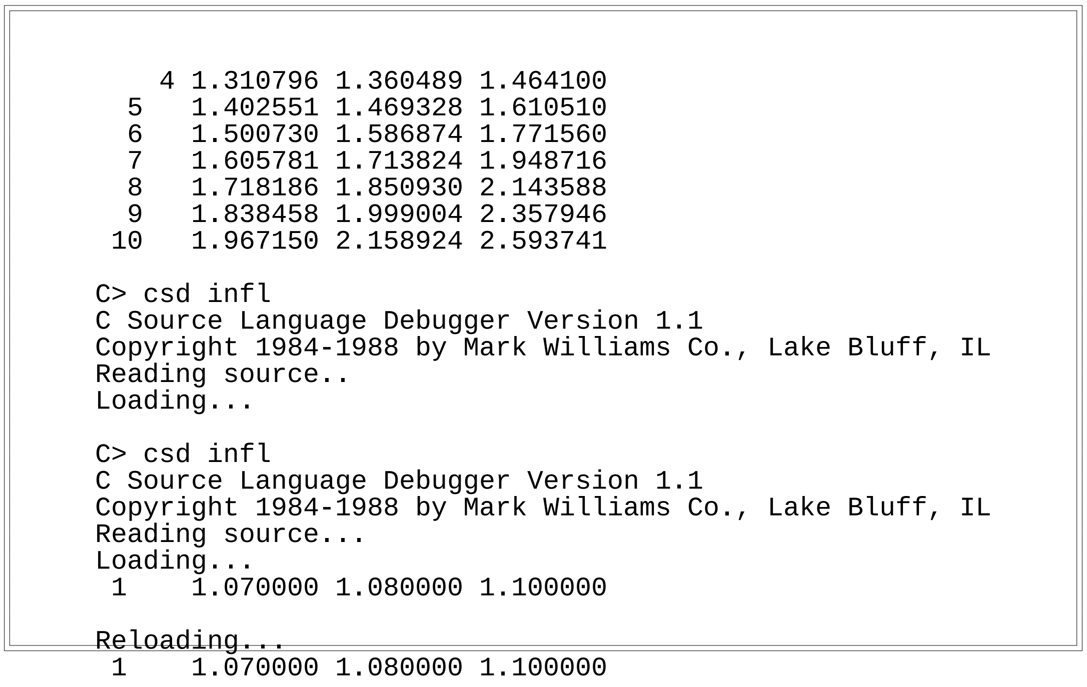
Again, press **<F4>** followed by **<return>**. **csd** runs to the next executable line in the program, which is:

```
w1 = 1.0;
```

Now, single-step through **infl.c**, pressing **<F4>** and **<return>**; stop after the first iteration of the **for** loop: your cursor will be positioned at the line

```
w1 *= 1.07;
```

for the second time since you began single-stepping. This is the point at which the next iteration of the **for** loop would begin. Press the **<F7>** key, and you will see the program output to this point. Your screen will appear as follows:



```
      4 1.310796 1.360489 1.464100
    5   1.402551 1.469328 1.610510
    6   1.500730 1.586874 1.771560
    7   1.605781 1.713824 1.948716
    8   1.718186 1.850930 2.143588
    9   1.838458 1.999004 2.357946
   10   1.967150 2.158924 2.593741

C> csd infl
C Source Language Debugger Version 1.1
Copyright 1984-1988 by Mark Williams Co., Lake Bluff, IL
Reading source..
Loading...

C> csd infl
C Source Language Debugger Version 1.1
Copyright 1984-1988 by Mark Williams Co., Lake Bluff, IL
Reading source...
Loading...
 1    1.070000 1.080000 1.100000

Reloading...
 1    1.070000 1.080000 1.100000
```

Once you single-stepped to the end of the **for** loop, **infl** generated one line of output, which you can see at the bottom of the screen.

If you wish, you can single-step through portions of a program. For example, you can locate the area in which a program goes astray by setting tracepoints; then, you can single-step through that area to find the exact line on which the problem occurs. Press **<F8>** to return to the source window.

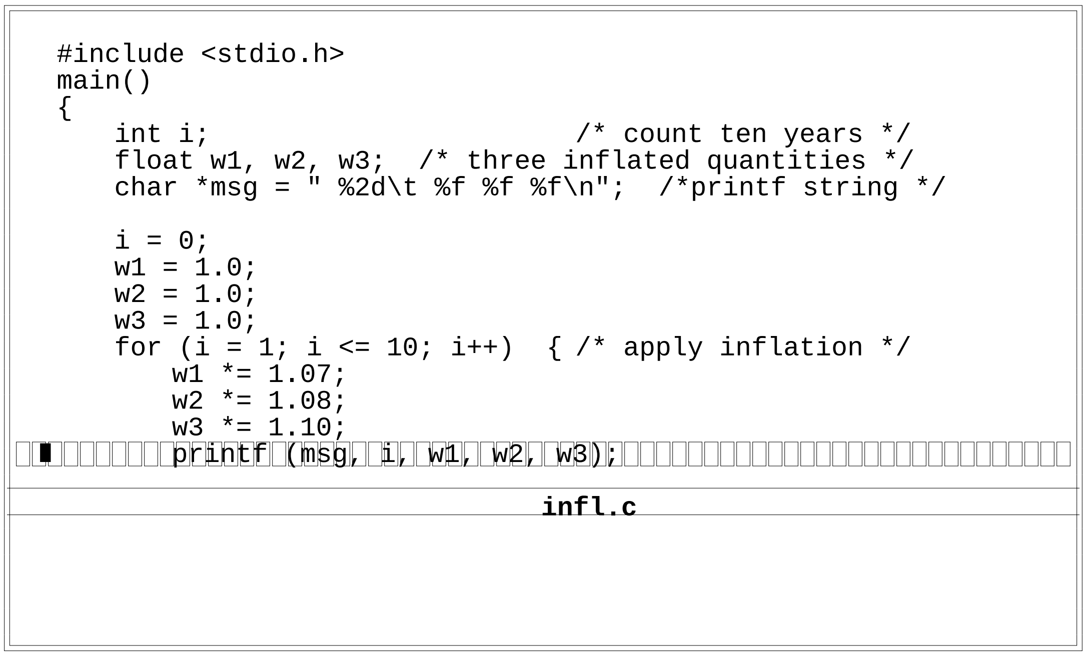
Displaying variables

csd lets you type expressions from your program in order to display the value of variables. You can evaluate any legal C expression, even function calls.

Local variables are declared at the beginning of a function. The local variable you wish to evaluate must be defined within the current scope. The *scope* of a local variable is the part of the program between the braces '{ }' of the function in which that variable is declared. Global, or *external*, variables are available from within any function's scope because they are defined outside of all functions.

Before entering local variables into **csd**'s evaluation window, you must first position the cursor in the source window so that it is between the braces of the function in which the variable is defined. Then switch to the evaluation window, and type in the variable you wish to see.

To illustrate how to display the value of variables and expressions, run the program to the end by typing **<F4>** followed by <End> . Now, position the cursor at the **printf** statement and type **<F3>** to set a tracepoint. Type **<F4>**, then **<F3>**: this tells **csd** to execute until it reaches the tracepoint. The screen now appears as follows:



```
#include <stdio.h>
main()
{
   int i;                       /* count ten years */
   float w1, w2, w3;  /* three inflated quantities */
   char *msg = " %2d\t %f %f %f\n";  /*printf string */

   i = 0;
   w1 = 1.0;
   w2 = 1.0;
   w3 = 1.0;
   for (i = 1; i <= 10; i++)  { /* apply inflation */
      w1 *= 1.07;
      w2 *= 1.08;
      w3 *= 1.10;
      printf (msg, i, w1, w2, w3);
                              infl.c
```

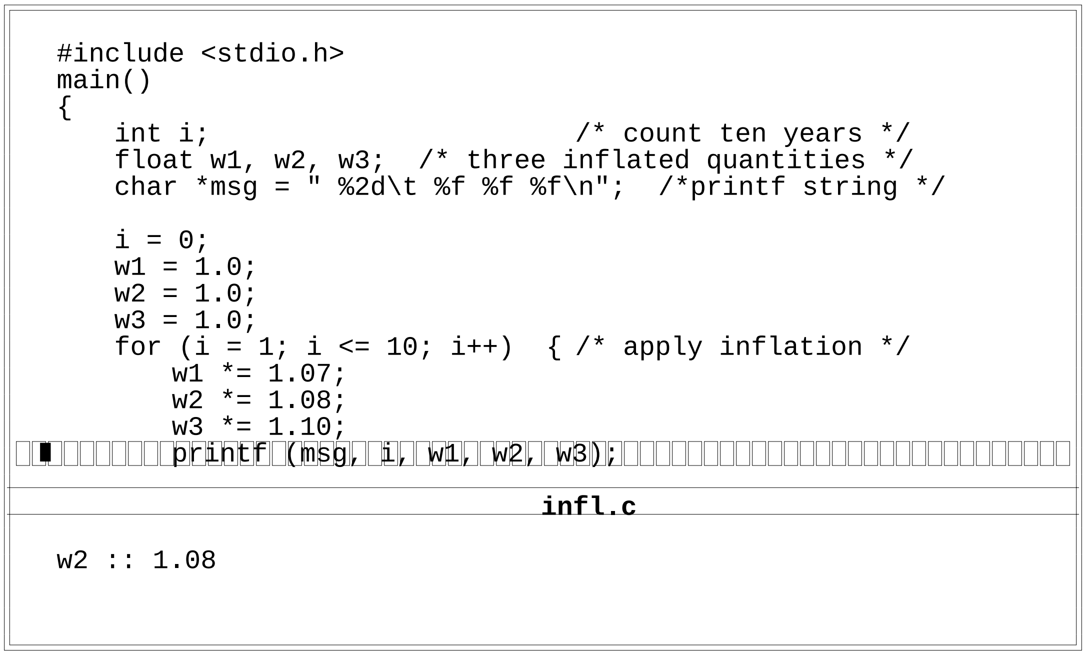
Now, switch to the evaluation window by pressing the **<F9>** key. The cursor is now flashing at the line below the reverse video line that says **infl.c**; this area of the screen is the evaluation window.

To examine the value of variable **w2**, type:

```
w2
```

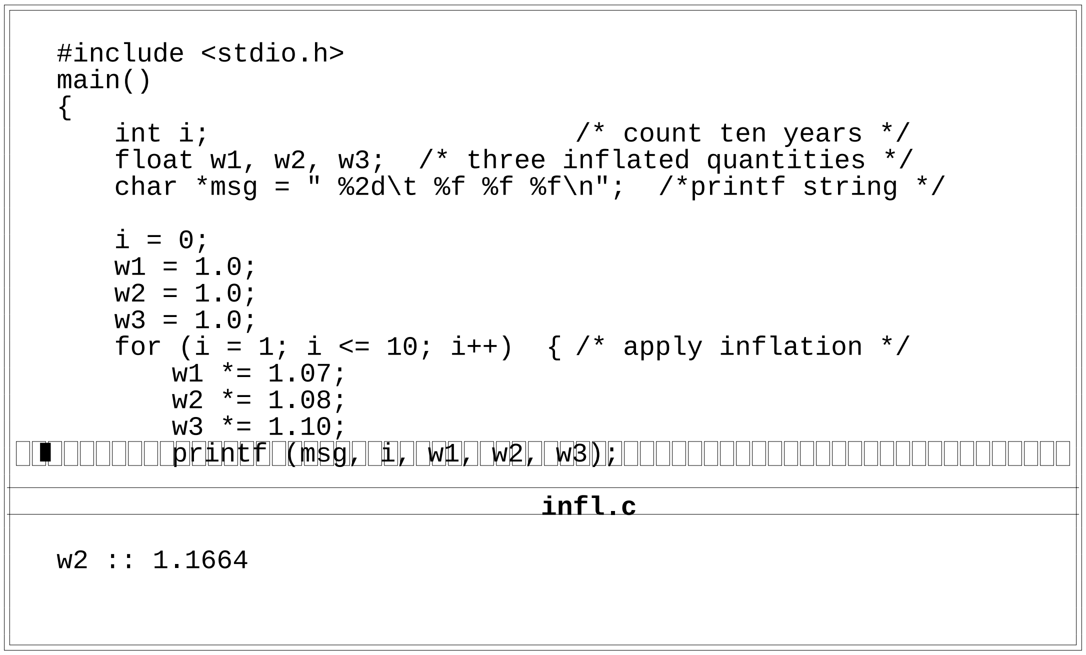
If you make a mistake typing in the evaluation window, use the backspace key to delete the error. You can delete an entire line by pressing **<ctrl-U>** as long as you have not yet typed **<return>**. When you have deleted the mistake, retype the expression.

As soon as you type **w2** followed by **<return>**, the screen will appear as follows:



```
#include <stdio.h>
main()
{
   int i;                        /* count ten years */
   float w1, w2, w3;  /* three inflated quantities */
   char *msg = " %2d\t %f %f %f\n";  /*printf string */

   i = 0;
   w1 = 1.0;
   w2 = 1.0;
   w3 = 1.0;
   for (i = 1; i <= 10; i++)  { /* apply inflation */
      w1 *= 1.07;
      w2 *= 1.08;
      w3 *= 1.10;
      printf (msg, i, w1, w2, w3);
                             infl.c

w2 :: 1.08
```

Notice that the value of **w2** is written after two colons, '::'. Typing **<F4>** and **<F3>** causes the program to cycle through the **for** loop once more; as you can see, the value of **w2** changes to reflect this further execution of the program. The screen now appears as follows:



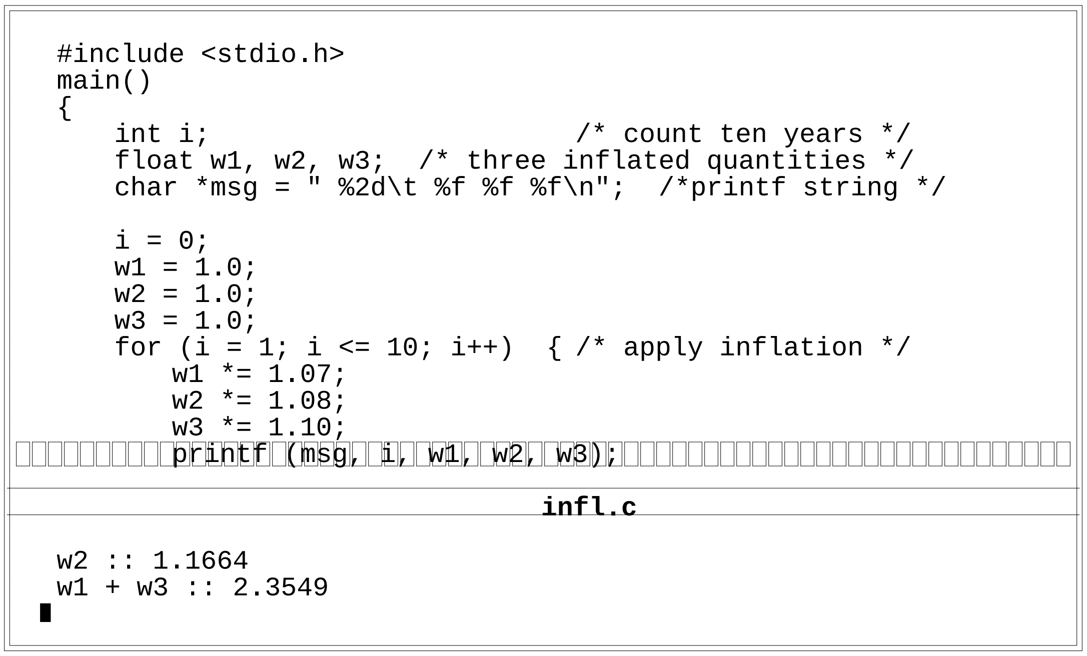
```
#include <stdio.h>
main()
{
    int i;                          /* count ten years */
    float w1, w2, w3;  /* three inflated quantities */
    char *msg = " %2d\t %f %f %f\n";  /*printf string */

    i = 0;
    w1 = 1.0;
    w2 = 1.0;
    w3 = 1.0;
    for (i = 1; i <= 10; i++)  { /* apply inflation */
        w1 *= 1.07;
        w2 *= 1.08;
        w3 *= 1.10;
▮       printf (msg, i, w1, w2, w3);
                              infl.c

w2 :: 1.1664
```

Remember that all evaluation expressions in the current scope are re-evaluated in the evaluation window whenever the debugger stops execution: whether for a traced statement, a traced expression, the end of the program, or for single-stepped execution.

You can evaluate all legal C expressions, even those that you did not anticipate when you wrote the program. To see how this works, return to the evaluation window (if your cursor is not there) by typing **<F9>**. Now, enter the following expression by typing:

```
w1 + w3
```

The screen will show:



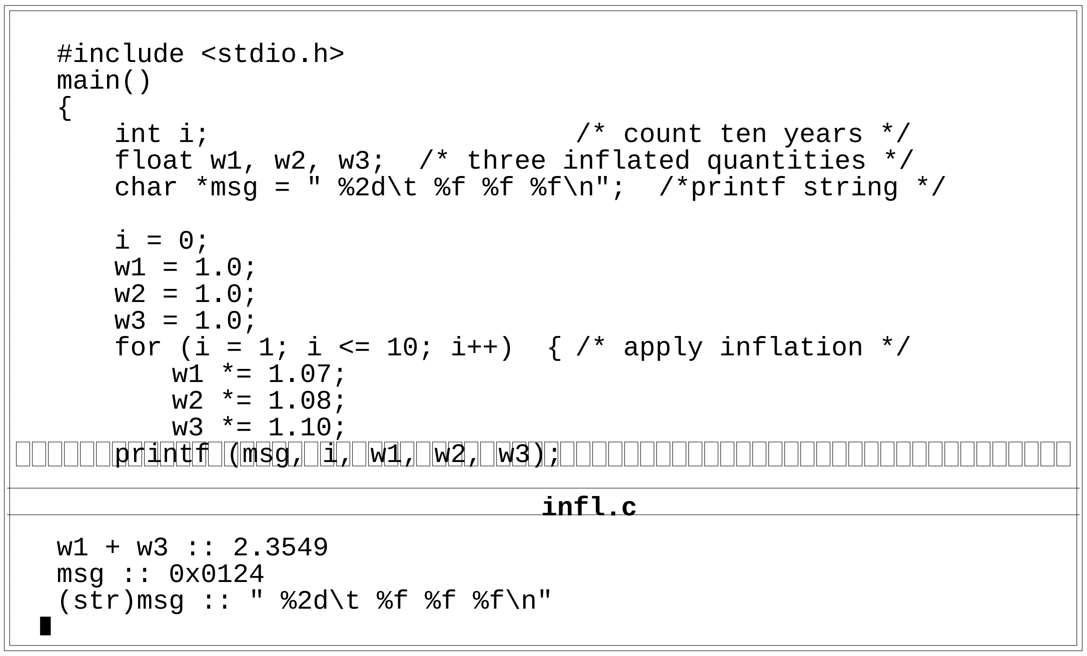
```
#include <stdio.h>
main()
{
   int i;                       /* count ten years */
   float w1, w2, w3;  /* three inflated quantities */
   char *msg = " %2d\t %f %f %f\n";  /*printf string */

   i = 0;
   w1 = 1.0;
   w2 = 1.0;
   w3 = 1.0;
   for (i = 1; i <= 10; i++)  { /* apply inflation */
      w1 *= 1.07;
      w2 *= 1.08;
      w3 *= 1.10;
      printf (msg, i, w1, w2, w3);
                                   infl.c

w2 :: 1.1664
w1 + w3 :: 2.3549
```

csd displays the results of adding **w1** with **w3** without your having to edit your program and recompile.

You can also show the value of a string pointer as a string, rather than as the address of the first character of the string, which is the way your program understands it. To do so, use **(str)** to *cast* the variable to a string type. For example, to see the value of the variable **msg** in both its forms, type the following:

```
msg
(str)msg
```

The screen will show:



```
#include <stdio.h>
main()
{
    int i;                      /* count ten years */
    float w1, w2, w3;  /* three inflated quantities */
    char *msg = " %2d\t %f %f %f\n";  /*printf string */

    i = 0;
    w1 = 1.0;
    w2 = 1.0;
    w3 = 1.0;
    for (i = 1; i <= 10; i++)  { /* apply inflation */
        w1 *= 1.07;
        w2 *= 1.08;
        w3 *= 1.10;
    printf (msg, i, w1, w2, w3);
                              infl.c
w1 + w3 :: 2.3549
msg :: 0x0124
(str)msg :: " %2d\t %f %f %f\n"
```

The value shown for **msg**, which may be different on your screen, is the address of the first character of its string: that of **(str)msg** is the string itself.

Note that if you enter an invalid variable in the evaluation window, **csd** prints an error message; you will not be able to continue debugging until the invalid variable is removed.

Getting help

Now that you have walked through a simple program and tried **csd**'s basic debugging features, try **csd** on a program of your own. At any time while you are using **csd**, you can get on-line help with its functions by typing the *help* key, **<F6>**.

Try typing **<F6>**. **csd** displays some general help information. Now, type any of the keys listed at the bottom of the general help screen. **csd** displays help information for the function you selected. For example, while the general help screen is displayed, pressing **<F3>** will show the help screen that tells you how to trace statements and expressions.

If **csd** gives you the error message

> *helpfile*.**hlp**: cannot open

instead of a help screen, it means that **csd** cannot find the help files in the current *path*. Use the -H option to tell **csd** where to find its help files. See section 6, *Commands reference*, for complete information about setting paths for **csd**.

When you are finished with the help feature, type **<F6>** again. The help menu will disappear, and the source window will be restored.

Finally, type **<Shift-F1>** to exit **csd** and return to .

Where to go from here

This section demonstrated the basics of **csd** on a simple program. It showed how to load a program and display its source code and output. It also discussed how to execute a program a line at a time or up to the next tracepoint, and how to examin the value of variables by typing an expression into the evaluation window.

The next section, *Advanced features*, will expand on the uses of these features. It will also demonstrate **csd**'s more powerful capabilities, such as its ability to trace expressions and function calls within the evaluation window.