Using GDB

GDB is a command-line debugger that you may find useful for finding bugs in your program. To use GDB effectively, you need to compile your program with the -g flag to gcc, g++ or fpc. This includes extra debugging information which helps gdb give useful information. You should also turn off all optimisation flags (eg, -O2) whilst debugging.

To debug your program under GDB, you need to type gdb ./myprog

The following list gives a quick reference for the most common gdb commands. A sample gdb session is shown overleaf. Pressing <enter> in GDB will execute the last command typed.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>Runs the program with gdb.</td>
</tr>
<tr>
<td>step</td>
<td>Executes the program until the next line in your source code is encountered. This will also step through subfunctions in your source code.</td>
</tr>
<tr>
<td>next</td>
<td>Same as above, but does not step through subfunctions.</td>
</tr>
<tr>
<td>print &lt;expression&gt;</td>
<td>Print the value of the expression. This may be a variable name, or other C expression (eg, my_array[13]*2)</td>
</tr>
<tr>
<td>watch &lt;expression&gt;</td>
<td>Display the value of the expression every time your program stops (eg, after a step, next, crash or breakpoint being hit).</td>
</tr>
<tr>
<td>backtrace</td>
<td>Will show the function calls that your program is currently nested within.</td>
</tr>
<tr>
<td>break &lt;line number</td>
<td>function&gt;</td>
</tr>
<tr>
<td>kill</td>
<td>Stops your program (you'll need to run it again if you want to continue debugging)</td>
</tr>
<tr>
<td>continue</td>
<td>Continue execution of your program until the next breakpoint, exit or crash.</td>
</tr>
<tr>
<td>list</td>
<td>Show the code that surrounds the line currently being executed.</td>
</tr>
<tr>
<td>quit</td>
<td>Quits from GDB</td>
</tr>
</tbody>
</table>

If your program is looping infinitely, you can press Ctrl+C to interrupt your program and see what is being executed.

Frontends to GDB exist, such as DDD, but are not covered here. They are slightly more intuitive to use.

printf is also a very valuable debugging tool, but be sure to turn off or remove your debugging output before submitting!
A sample GDB session is shown below. Text that you type is in bold.

n113112:~> gcc -Wall -o test -g test.c
n113112:~> gdb ./out
GNU gdb 6.3-debian
[...]
(gdb) run
Starting program: /tmp/test

hi
Program received signal SIGSEGV, Segmentation fault.
0x0804836e in main () at test.c:8
3               return *a;
(gdb) backtrace
#0  0x08048369 in do_something_silly () at test.c:8
#1  0x0804835f in do_something () at test.c:3
#2  0x08048382 in main () at test.c:11
(gdb) print a
$1 = (int *) 0x0
(gdb) kill
Kill the program being debugged? (y or n) y
(gdb)

(gdb) break do_something_silly
Breakpoint 1 at 0x8048397: file test.c, line 7.
(gdb) run
Starting program: /tmp/test

Breakpoint 1, do_something_silly () at test.c:7
7               printf("hi\n");
(gdb) step
hi
8               return *a;
(gdb) <enter>

Program received signal SIGSEGV, Segmentation fault.
0x080483a8 in do_something_silly () at test.c:8
8               return *a;
(gdb) list
3                   return do_something_silly();
4
5
6                   int do_something_silly() {
7                       printf("hi\n");
8                   return *a;
9                   }
10
11      int main() {
12                      return do_something();
(gdb) q
The program is running. Exit anyway? (y or n) y
n113112:~>