

EJERCICIO 1:

Para este ejercicio, lo primero que debemos hacer es ejecutar la orden que nos dice el PDF, pero, a primera vista, no será posible depurar el ejecutable, ya que no hemos añadido “-g”. Efectivamente, si compilamos así, y después intentamos aplicar el comando “list”, no ocurre nada.

```
sesion08 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ g++ main.cpp factorial.cpp hello.cpp -o ejecutable
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ gdb ejecutable
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejecutable...(no se encontraron símbolos de depuración)hecho.
(gdb) list
No hay tabla de símbolos cargada. Use la orden «file».
(gdb) q
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$
```

Por tanto, para solucionar esto, basta con añadir “-g” a la hora de compilar nuestros “.cpp”. Además, haremos uso del comando “list” y “quit”, para mostrar el código desde el depurador y salir del depurador respectivamente.

```
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ g++ -g main.cpp factorial.cpp hello.cpp -o ejecutable
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ gdb ejecutable
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejecutable...hecho.
(gdb) list
1      #include <iostream>
2      #include "functions.h"
3
4      using namespace std;
5
6      int main(){
7          print_hello();
8          cout << endl;
9          cout << "The factorial of 7 is " << factorial(7) << endl;
10         return 0;
(gdb) quit
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$
```

EJERCICIO 2:

Para ejecutar la función factorial, bastaría con poner un punto de ruptura con la orden "break" y añadiendo el nombre de nuestra función "factorial" para después ir avanzando en la ejecución de esta función con el comando "step".

```
sesion08
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
davidms_83@davidMSHp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ gdb ejecutable
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejecutable...hecho.
(gdb) break factorial
Punto de interrupción 1 at 0xa67: file factorial.cpp, line 4.
(gdb) run
Starting program: /home/davidms_83/Documentos/FS/PRACTICAS/PRACTICA8/sesion08/ejecutable
Hello World!

Breakpoint 1, factorial (n=7) at factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=6) at factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=5) at factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=4) at factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=3) at factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
```

```
> ✂
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=6) at factorial.cpp:4
4          if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=5) at factorial.cpp:4
4          if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=4) at factorial.cpp:4
4          if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=3) at factorial.cpp:4
4          if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

Breakpoint 1, factorial (n=2) at factorial.cpp:4
4          if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

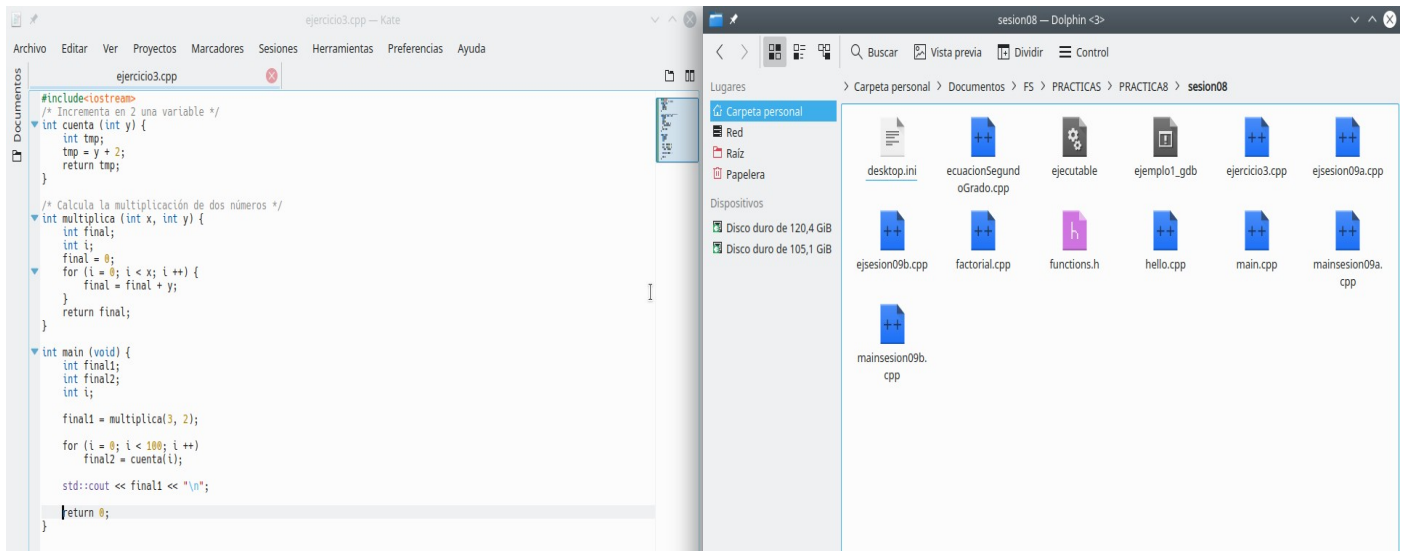
Breakpoint 1, factorial (n=1) at factorial.cpp:4
4          if(n!=1){
(gdb) step
7          else return 1;
(gdb) step
8          }
(gdb) step
8          }
(gdb) step
8          }
(gdb) q
Una sesión de depuración está activa.

Inferior 1 [process 10993] will be killed.

¿Salir de cualquier modo? (y or n) y
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$
```

EJERCICIO 3

Lo primero para realizar este ejercicio, es copiar el código del PDF en un archivo de texto vacío con extensión “.cpp”. Después lo compilaremos para crear un ejecutable (incluyendo “-g”). Lo he guardado bajo el nombre “ejercicio3.cpp”, en la carpeta “sesion08”, donde estamos realizando todos los ejercicios.



Para realizar el ejercicio, lo más conveniente es hacer un punto de ruptura en una línea donde todavía no se haya ejecutado el bucle que le asigna valores a la variable “final2”, pero que esté después de la declaración de esta. Para ver que línea escoger, hacemos “list 1,30”, y así se nos mostrarían las 30 primeras líneas de nuestro “ejercicio3.cpp”. Para mostrar el valor de la variable, aplicamos “display final2”.

(Adjunto imagen en la siguiente página).


```
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
davidms_83@davidMSHp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ g++ -g ejercicio3.cpp -o ejercicio3
davidms_83@davidMSHp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ gdb ejercicio3
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejercicio3...hecho.
(gdb) list 1,30
1      #include<iostream>
2      /* Incrementa en 2 una variable */
3      int cuenta (int y) {
4          int tmp;
5          tmp = y + 2;
6          return tmp;
7      }
8
9      /* Calcula la multiplicación de dos números */
10     int multiplica (int x, int y) {
11         int final;
12         int i;
13         final = 0;
14         for (i = 0; i < x; i++) {
15             final = final + y;
16         }
17         return final;
18     }
19
20     int main (void) {
21         int final1;
22         int final2;
23         int i;
24
25         final1 = multiplica(3, 2);
26
27         for (i = 0; i < 100; i++)
28             final2 = cuenta(i);
29
30         std::cout << final1 << "\n";
(gdb) break 23
Punto de interrupción 1 at 0x8fe: file ejercicio3.cpp, line 23.
(gdb) run
Starting program: /home/davidms_83/Documentos/FS/PRACTICAS/PRACTICA8/sesion08/ejercicio3

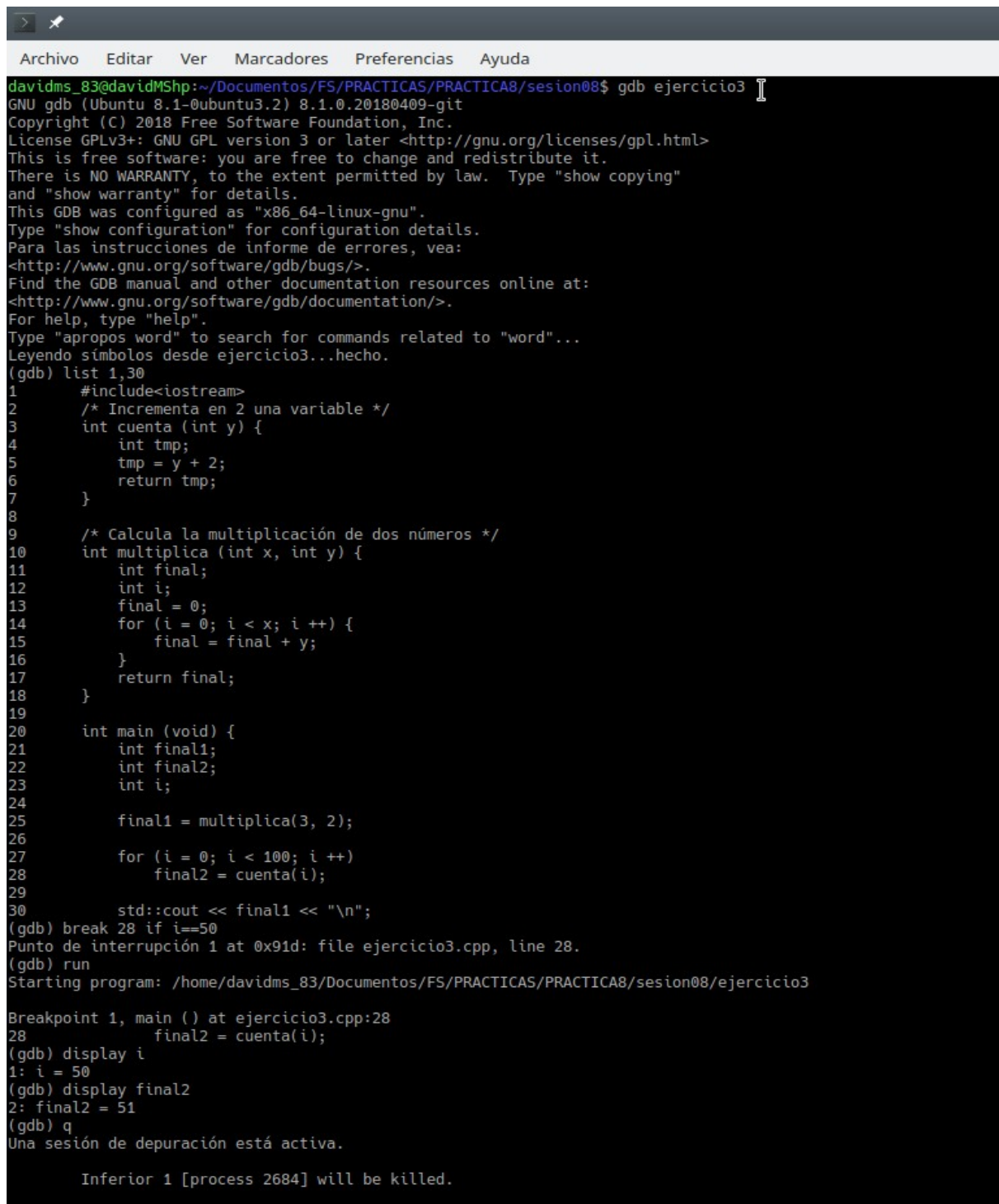
Breakpoint 1, main () at ejercicio3.cpp:25
25         final1 = multiplica(3, 2);
(gdb) display final2
1: final2 = 0
(gdb) q
Una sesión de depuración está activa.

Inferior 1 [process 13700] will be killed.
¿Salir de cualquier modo? (y or n) y
```

Como se puede observar en la imagen, la variable "i" vale 0.

EJERCICIO 4

Este ejercicio es similar al anterior. La única diferencia es el uso de puntos de ruptura condicionales. Para poner un punto de ruptura justo cuando "i" valga 50, debemos escribir lo siguiente en GDB, "break (línea donde se hace la asignación a final2 dentro del bucle) if i == 50". Después para ver el valor de ambas variables bastará con hacer "display i" y "display final2".



```
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
davidms_83@davidMSHp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ gdb ejercicio3
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejercicio3...hecho.
(gdb) list 1,30
1      #include<iostream>
2      /* Incrementa en 2 una variable */
3      int cuenta (int y) {
4          int tmp;
5          tmp = y + 2;
6          return tmp;
7      }
8
9      /* Calcula la multiplicación de dos números */
10     int multiplica (int x, int y) {
11         int final;
12         int i;
13         final = 0;
14         for (i = 0; i < x; i++) {
15             final = final + y;
16         }
17         return final;
18     }
19
20     int main (void) {
21         int final1;
22         int final2;
23         int i;
24
25         final1 = multiplica(3, 2);
26
27         for (i = 0; i < 100; i++)
28             final2 = cuenta(i);
29
30         std::cout << final1 << "\n";
(gdb) break 28 if i==50
Punto de interrupción 1 at 0x91d: file ejercicio3.cpp, line 28.
(gdb) run
Starting program: /home/davidms_83/Documentos/FS/PRACTICAS/PRACTICA8/sesion08/ejercicio3

Breakpoint 1, main () at ejercicio3.cpp:28
28         final2 = cuenta(i);
(gdb) display i
1: i = 50
(gdb) display final2
2: final2 = 51
(gdb) q
Una sesión de depuración está activa.

Inferior 1 [process 2684] will be killed.
```

Como podemos ver, en la imagen anteriormente adjuntada, la "i" vale 50 y "final2", vale 51.

EJERCICIO 5

Para este ejercicio, lo primero que haremos será declarar un punto de ruptura en la función "cuenta", en la línea donde se hace el "return", si es la última iteración, es decir, si "y=99" y cambiamos "temp", mediante "set variable" para que valga 1001. Para ver si el valor de "final2" tras todas las iteraciones del bucle supera los 1000, declararemos un punto de ruptura justo después del bucle, y haremos "display final2".

(Adjunto imágenes en las siguientes páginas).

```
> ✕
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
davidms_83@davidMSHp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$ gdb ejercicio3
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
Para las instrucciones de informe de errores, vea:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Leyendo símbolos desde ejercicio3...hecho.
(gdb) list 1,30
1      #include<iostream>
2      /* Incrementa en 2 una variable */
3      int cuenta (int y) {
4          int tmp;
5          tmp = y + 2;
6          return tmp;
7      }
8
9      /* Calcula la multiplicación de dos números */
10     int multiplica (int x, int y) {
11         int final;
12         int i;
13         final = 0;
14         for (i = 0; i < x; i++) {
15             final = final + y;
16         }
17         return final;
18     }
19
20     int main (void) {
21         int final1;
22         int final2;
23         int i;
24
25         final1 = multiplica(3, 2);
26
27         for (i = 0; i < 100; i++)
28             final2 = cuenta(i);
29
30         std::cout << final1 << "\n";
(gdb) break 6 if y==99
Punto de interrupción 1 at 0x8c0: file ejercicio3.cpp, line 6.
(gdb) break 30
Punto de interrupción 2 at 0x930: file ejercicio3.cpp, line 30.
(gdb) run
Starting program: /home/davidms_83/Documentos/FS/PRACTICAS/PRACTICA8/sesion08/ejercicio3

Breakpoint 1, cuenta (y=99) at ejercicio3.cpp:6
6          return tmp;
(gdb) set variable tmp=1001
(gdb) next
7      }
(gdb) next
main () at ejercicio3.cpp:27
27         for (i = 0; i < 100; i++)
(gdb) next
```



```
> ✕
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
3      int cuenta (int y) {
4          int tmp;
5          tmp = y + 2;
6          return tmp;
7      }
8
9      /* Calcula la multiplicación de dos números */
10     int multiplica (int x, int y) {
11         int final;
12         int i;
13         final = 0;
14         for (i = 0; i < x; i++) {
15             final = final + y;
16         }
17         return final;
18     }
19
20     int main (void) {
21         int final1;
22         int final2;
23         int i;
24
25         final1 = multiplica(3, 2);
26
27         for (i = 0; i < 100; i++)
28             final2 = cuenta(i);
29
30         std::cout << final1 << "\n";
(gdb) break 6 if y==99
Punto de interrupción 1 at 0x8c0: file ejercicio3.cpp, line 6.
(gdb) break 30
Punto de interrupción 2 at 0x930: file ejercicio3.cpp, line 30.
(gdb) run
Starting program: /home/davidms_83/Documentos/FS/PRACTICAS/PRACTICA8/sesion08/ejercicio3

Breakpoint 1, cuenta (y=99) at ejercicio3.cpp:6
6          return tmp;
(gdb) set variable tmp=1001
(gdb) next
7      }
(gdb) next
main () at ejercicio3.cpp:27
27         for (i = 0; i < 100; i++)
(gdb) next

Breakpoint 2, main () at ejercicio3.cpp:30
30         std::cout << final1 << "\n";
(gdb) display final2
1: final2 = 1001
(gdb) next
6
32         return 0;
1: final2 = 1001
(gdb) next
33     }
1: final2 = 1001
(gdb) q
Una sesión de depuración está activa.

Inferior 1 [process 12099] will be killed.

¿Salir de cualquier modo? (y or n) y
davidms_83@davidMShp:~/Documentos/FS/PRACTICAS/PRACTICA8/sesion08$
```