## Ejercicio 9.1.

**Compile los archivos main.cpp factorial.cpp hello.cpp y genere un ejecutable con el nombre ejemplo9.1. Lance gdb con dicho ejemplo y ejecútelo dentro del depurador. Describa la información que ofrece.**

```
juanka1995@juanka1995-VBox:~/modulo2/sesion09$ g++ -g main.cpp hello.cpp factori
al.cpp -o ejemplo9.1
juanka1995@juanka1995-VBox:~/modulo2/sesion09$ ls -l ejemplo9.1
-rwxrwxr-x 1 juanka1995 juanka1995 27365 ene 12 11:39 ejemplo9.1
```

```
juanka1995@juanka1995-VBox:~/modulo2/sesion09$ gdb ejemplo9.1
GNU gdb (Ubuntu 7.7-0ubuntu3.1) 7.7
Copyright (C) 2014 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 ejemplo9.1...hecho.
(gdb) run
Starting program: /home/juanka1995/modulo2/sesion09/ejemplo9.1
Traceback (most recent call last):
  File "/usr/share/gdb/auto-load/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.19-gd
b.py", line 63, in <module>
    from libstdcxx.v6.printers import register_libstdcxx_printers
ImportError: No module named 'libstdcxx'
Hello World!
The factorial of 7 is 5040
[Inferior 1 (process 2599) exited normally]
```

Nos informa de que la ejecución de nuestro programa se ha llevado a cabo correctamente y del PID del proceso que utilizaba.

## Ejercicio 9.2.

**Usando la orden list muestre el código del programa principal y el de la función factorial utilizados en el ejercicio 9.1 (para ello utilice la orden help list).**

```
(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) list factorial
1       #include "functions.h"
2
3       int factorial(int n){
4           if(n!=1){
5               return(n * factorial(n-1));
6           }
7           else return 1;
8       }
```

## Ejercicio 9.3.

**Ponga un punto de ruptura asociado a cada línea del programa fuente mainsesion09a.cpp donde aparezca el comentario /* break */. Muestre información de todas las variables que se estén usando cada vez que en la depuración se detenga la ejecución. Muestre la información del contador de programa mediante $pc y el de la pila con $sp.**

```
(gdb) break 15
Punto de interrupción 1 at 0x4007dd: file mainsesion09a.cpp, line 15.
(gdb) break 29
Punto de interrupción 2 at 0x4007fc: file mainsesion09a.cpp, line 29.
(gdb) break 42
Punto de interrupción 3 at 0x40081b: file mainsesion09a.cpp, line 42.
(gdb) break 47
Punto de interrupción 4 at 0x400836: file mainsesion09a.cpp, line 47.
```

## Ejercicio 9.4.

**Indique las órdenes necesarias para ver el valor de las variables final1 y final2 del programa generado en el ejercicio anterior en los puntos de ruptura correspondientes tras un par de iteraciones en el bucle for. Indique la orden para obtener el código ensamblador de la zona depurada.**

Gdb mainsesion09a.cpp
break 47
run
print final1
print final2
c
c
c
print final1
print final2

disassemble

## Ejercicio 9.5.

**Considerando la depuración de los ejercicios anteriores, elimine todos los puntos de ruptura salvo el primero.**

```
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004007dd in cuenta(int)
                                                   at mainsesion09a.cpp:15
2       breakpoint     keep y   0x00000000004007fc in multiplica(int, int)
                                                   at mainsesion09a.cpp:29
3       breakpoint     keep y   0x000000000040081b in main()
                                                   at mainsesion09a.cpp:42
4       breakpoint     keep y   0x0000000000400836 in main()
                                                   at mainsesion09a.cpp:47
(gdb) delete 2
(gdb) delete 3
(gdb) delete 4
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004007dd in cuenta(int)
                                                   at mainsesion09a.cpp:15
```

## Ejercicio 9.7.

**Realice la depuración del programa ejecutable obtenido a partir del archivo fuente ejsesion09a.cpp. Utilizando gdb, trate de averiguar qué sucede y por qué no funciona. Intente arreglar el programa.**

G++ -g ejsesion09A.cpp -o ejsesion09A
gdb ejsesion09A
list 1,43

Analizando el codigo me dado cuenta que el error estaba en la linea numero 27. Su correción seria la siguiente:

tmp = suma(tmp, vector[i]);

## Ejercicio 9.8.

**Compile el programa mainsesion09b.cpp y genere un ejecutable con el nombre ejemplo9.8.**
**Ejecute gdb con dicho ejemplo y realice una ejecución depurada mediante la orden run.**
**Añada un punto de ruptura (breakpoint) en la línea donde se invoca a la función cuenta (se puede realizar tal y como se muestra en el ejemplo anterior o mediante el número de línea donde aparezca la llamada a esa función).**
**Realice 10 pasos de ejecución con step y otros 10 con next. Comente las diferencias.**

G++ -g mainsesion09b.cpp -o ejemplo9.8
gdb ejemplo9.8
break cuenta
info b

```
(gdb) s
17          return tmp;
(gdb) s
18      }
(gdb) s
main () at mainsesion09b.cpp:46
46          for (i = 0; i < 100; i ++)
(gdb) s
48              final2 = cuenta(i);
(gdb) s

Breakpoint 1, cuenta (y=1) at mainsesion09b.cpp:13
13          tmp = y + 2;
(gdb) s
17          return tmp;
(gdb) s
18      }
(gdb) s
main () at mainsesion09b.cpp:46
46          for (i = 0; i < 100; i ++)
(gdb) s
48              final2 = cuenta(i);
(gdb) s

Breakpoint 1, cuenta (y=2) at mainsesion09b.cpp:13
13          tmp = y + 2;
```

```
(gdb) n
17          return tmp;
(gdb) n
18      }
(gdb) n
main () at mainsesion09b.cpp:46
46          for (i = 0; i < 100; i ++)
(gdb) n
48              final2 = cuenta(i);
(gdb) n

Breakpoint 1, cuenta (y=3) at mainsesion09b.cpp:13
13          tmp = y + 2;
(gdb) n
17          return tmp;
(gdb) n
18      }
(gdb) n
main () at mainsesion09b.cpp:46
46          for (i = 0; i < 100; i ++)
(gdb) n
48              final2 = cuenta(i);
(gdb) n

Breakpoint 1, cuenta (y=4) at mainsesion09b.cpp:13
13          tmp = y + 2;
```

==== next (n) ====
Ejecuta la siguiente línea de programa. Si es una función, ejecuta la función entera (no entra en ella).
==== step (s) ====
Como next, pero si es una función entra en ella.

## Ejercicio 9.9.

**Depure el programa generado en el ejercicio anterior. Introduzca un punto de ruptura ( breakpoint ) dentro de la función cuenta. Usando la orden info frame, muestre la información del marco actual y del marco superior; vuelva al marco inicial y compruebe si ha cambiado algo.**

```
Breakpoint 1, cuenta (y=0) at mainsesion09b.cpp:13
13          tmp = y + 2;
(gdb) info frame
Stack level 0, frame at 0x7fffffffdee0:
 rip = 0x4007d4 in cuenta (mainsesion09b.cpp:13); saved rip = 0x400840
 called by frame at 0x7fffffffdf00
 source language c++.
 Arglist at 0x7fffffffded0, args: y=0
 Locals at 0x7fffffffded0, Previous frame's sp is 0x7fffffffdee0
 Saved registers:
  rbp at 0x7fffffffded0, rip at 0x7fffffffded8
(gdb) up
#1  0x0000000000400840 in main () at mainsesion09b.cpp:48
48              final2 = cuenta(i);
(gdb) info frame
Stack level 1, frame at 0x7fffffffdf00:
 rip = 0x400840 in main (mainsesion09b.cpp:48); saved rip = 0x7ffff7731ec5
 caller of frame at 0x7fffffffdee0
 source language c++.
 Arglist at 0x7fffffffdef0, args:
 Locals at 0x7fffffffdef0, Previous frame's sp is 0x7fffffffdf00
 Saved registers:
  rbp at 0x7fffffffdef0, rip at 0x7fffffffdef8
(gdb) down
#0  cuenta (y=0) at mainsesion09b.cpp:13
13          tmp = y + 2;
(gdb) info frame
Stack level 0, frame at 0x7fffffffdee0:
 rip = 0x4007d4 in cuenta (mainsesion09b.cpp:13); saved rip = 0x400840
 called by frame at 0x7fffffffdf00
 source language c++.
 Arglist at 0x7fffffffded0, args: y=0
 Locals at 0x7fffffffded0, Previous frame's sp is 0x7fffffffdee0
 Saved registers:
  rbp at 0x7fffffffded0, rip at 0x7fffffffded8
```

No observo ningún cambio.

## Ejercicio 9.10.

**Ponga un punto de ruptura en la línea 30 del programa utilizado en el ejercicio anterior (función multiplica) de tal forma que el programa se detenga cuando la variable final tenga como valor 8. Compruebe si se detiene o no y explique por qué.**

(gdb) **break 30 if final==8**
Punto de interrupción 1 at 0x4007fc: file mainsesion09b.cpp, line 30.
(gdb) **run**
Starting program: /home/juanka1995/modulo2/sesion09/ejemplo9.8
Traceback (most recent call last):
  File "/usr/share/gdb/auto-load/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.19-gdb.py", line 63, in <module>
    from libstdcxx.v6.printers import register_libstdcxx_printers
ImportError: No module named 'libstdcxx'
6
[Inferior 1 (process 3520) exited normally]

No se detiene debido a que solo existe un llamada a la función multiplica y se le pasan los valores 3 y 2 que al multiplicarlos dan 6 y nunca darán 8.

## Ejercicio 9.11.

**Pruebe el ejemplo anterior, ejecute después un continue y muestre el valor de la variable tmp. Todo haría indicar que el valor debiera ser 12 y sin embargo no es así, explique por qué.**

(gdb) **break 10**
Punto de interrupción 2 at 0x4007d4: file mainsesion09b.cpp, line 10.
(gdb) **run**
Starting program: /home/juanka1995/modulo2/sesion09/ejemplo9.8
Traceback (most recent call last):
  File "/usr/share/gdb/auto-load/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.19-gdb.py", line 63, in <module>
    from libstdcxx.v6.printers import register_libstdcxx_printers
ImportError: No module named 'libstdcxx'

Breakpoint 2, cuenta (y=0) at mainsesion09b.cpp:13
13          tmp = y + 2;
(gdb) **print tmp**
$1 = 3
(gdb) **set variable tmp=10**
(gdb) **print tmp**
$2 = 10
(gdb) **c**
Continuando.

Breakpoint 2, cuenta (y=1) at mainsesion09b.cpp:13
13          tmp = y + 2;
(gdb) **print tmp**
$3 = 2

La variable tmp es una variable interna de la función *cuenta*, cuyo valor cambia cada vez que ejecutamos una llamada a dicha función. Por eso cuando damos a continuar el valor cambia por que recibe otro valor distinto y machaca el valor 10 que habíamos asignado.