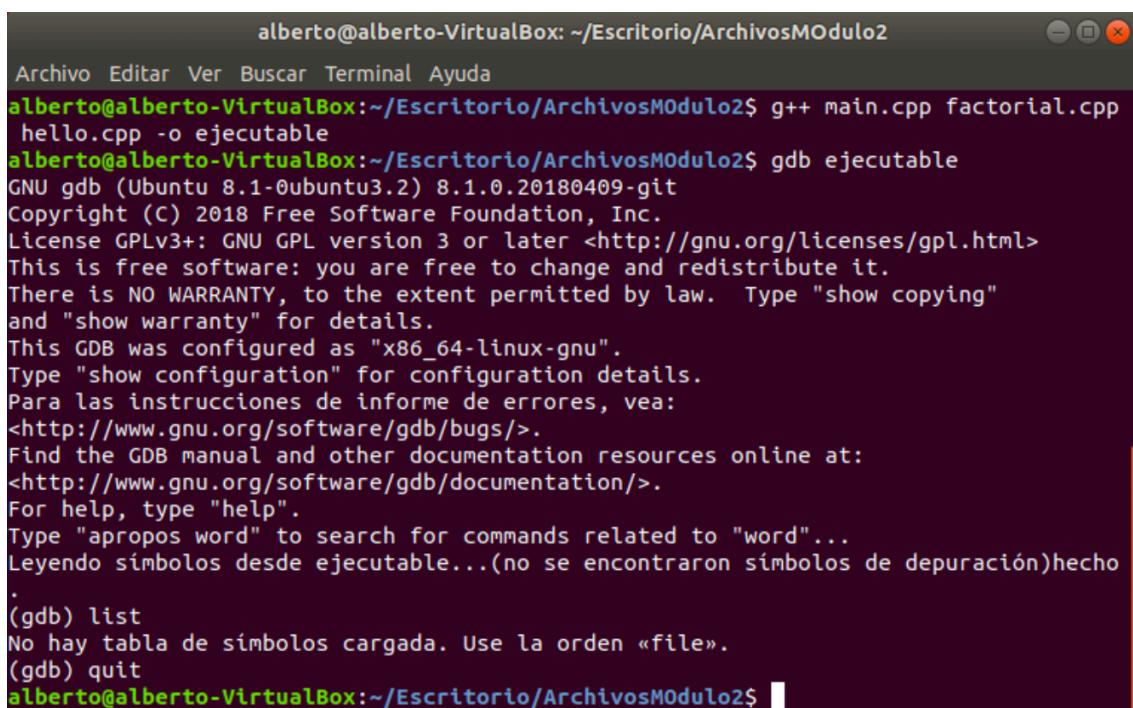


## PRACTICA 8 – ALBERTO LLAMAS GONZÁLEZ

**Ejercicio 1.** Utilizando los archivos disponibles en la carpeta de la sesión 8 (main.cpp, factorial.cpp y hello.cpp). Ejecuta la siguiente orden: `g++ main.cpp factorial.cpp hello.cpp -o ejecutable` Una vez compilado el código, utiliza la herramienta gdb para ver el código del programa principal. Describe qué sucede al intentar ver las líneas del código del main y cómo harías para ver el código por medio del depurador. Indica cómo se puede salir del depurador.

Ejecutemos la orden tal y como viene en el enunciado:



```
alberto@alberto-VirtualBox: ~/Escritorio/ArchivosMÓdulo2
Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMÓdulo2$ g++ main.cpp factorial.cpp
hello.cpp -o ejecutable
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMÓdulo2$ 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) quit
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMÓdulo2$
```

(Imagen 1.1)

Como podemos observar en la imagen 1.1 se ha ejecutado la orden `g++ main.cpp factorial.cpp hello.cpp -o ejecutable` y se ha procedido a ejecutar el depurador gdb. Sin embargo, al intentar ver el código del programa principal mediante la orden `list`, se puede apreciar cómo no es posible visualizar su contenido. Para poder ver el código mediante el depurador, es necesario añadir la opción ‘`-g`’ que permite añadir la información necesaria para poder depurar el programa. Podemos ver lo que ocurre en la imagen 1.2:

```
alberto@alberto-VirtualBox: ~/Escritorio/ArchivosM0dulo2
Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox:~/Escritorio/ArchivosM0dulo2$ g++ -g main.cpp factorial.cpp hello.cpp
-o ejecutable
alberto@alberto-VirtualBox:~/Escritorio/ArchivosM0dulo2$ 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
alberto@alberto-VirtualBox:~/Escritorio/ArchivosM0dulo2$
```

(Imagen 1.2)

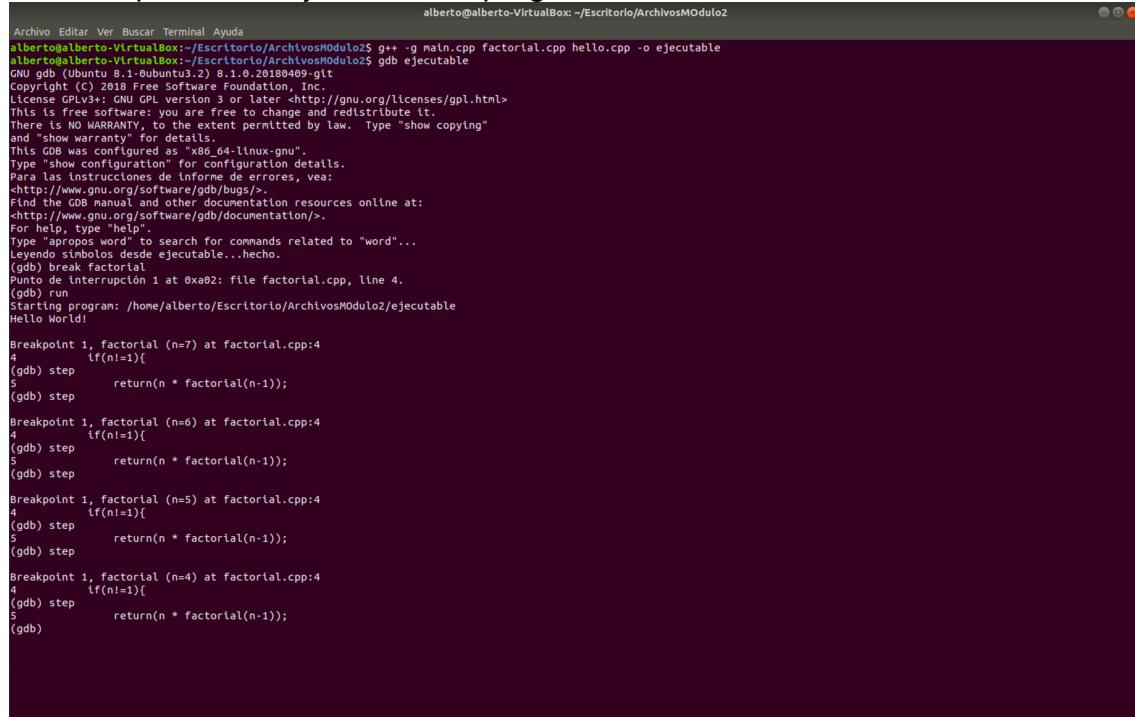
Como podemos ver en ambas imágenes (1.1 y 1.2), para salir del depurador se utiliza la orden *quit*. Para poder ver el contenido del *main*, también podemos utilizar la orden *list 1,10*, pero para ello, necesitamos conocer el número de líneas de código del *main*.

Lista de comandos:

- *g++ main.cpp factorial.cpp hello.cpp –o ejecutable*
- *gdb ejecutable*
- *list*
- *quit*
- *g++ -g main.cpp factorial.cpp hello.cpp –o ejecutable*

**Ejercicio 2.** ¿Cómo harías para ejecutar la función factorial del ejercicio anterior? Muestra una captura de pantalla indicando el resultado y cómo has llegado a él.

Veamos primero la ejecución del programa:



```

Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMODOulo2$ g++ -g main.cpp factorial.cpp hello.cpp -o ejecutable
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMODOulo2$ gdb ejecutable
GNU gdb (Ubuntu 8.1.0-0ubuntu1.2) 8.1.0.20180409-glt
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"...
Leiendo símbolos desde ejecutable...hecho.
(gdb) break factorial
Punto de interrupción 1 en 0x0000555555555554: file factorial.cpp, línea 4.
(gdb) run
Starting program: /home/alberto/Escritorio/ArchivosMODOulo2/ejecutable
Hello World!

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

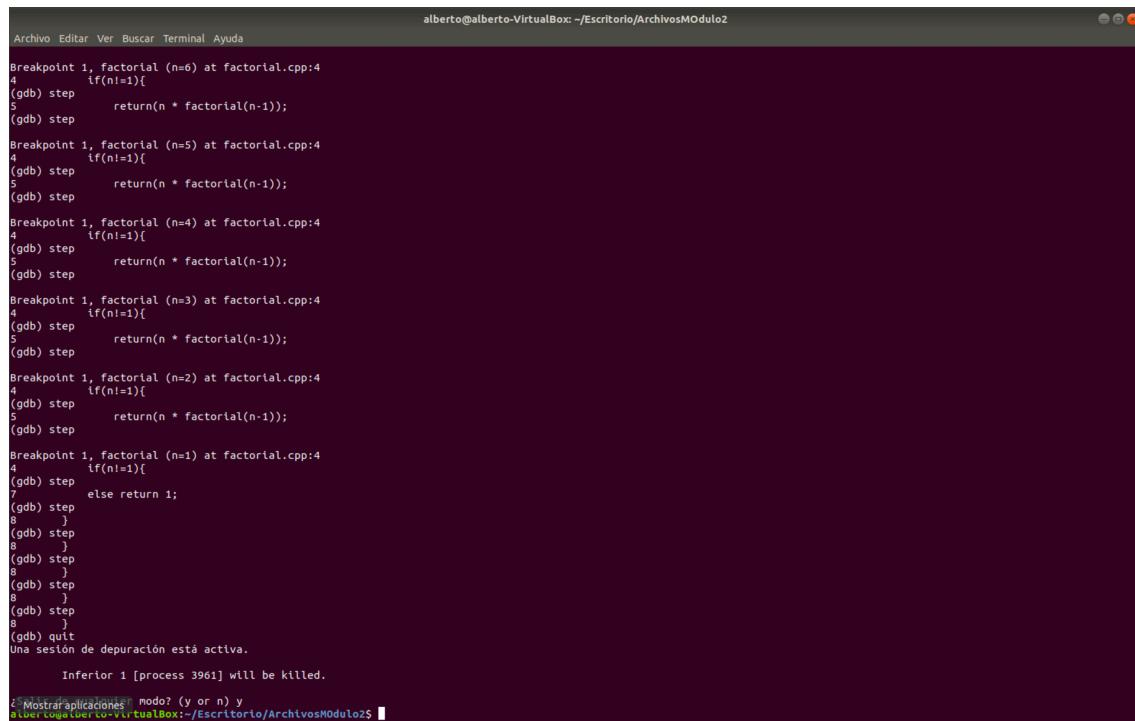
Breakpoint 1, factorial (n=6) en factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

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

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

```

(Imagen 2.1)



```

Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox: ~/Escritorio/ArchivosMODOulo2
Breakpoint 1, factorial (n=6) en factorial.cpp:4
4      if(n!=1){
(gdb) step
5          return(n * factorial(n-1));
(gdb) step

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

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

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

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

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

Inferior 1 [process 3961] will be killed.

$ alberto@alberto-VirtualBox:~/Escritorio/ArchivosMODOulo2$ 

```

(Imagen 2.2)

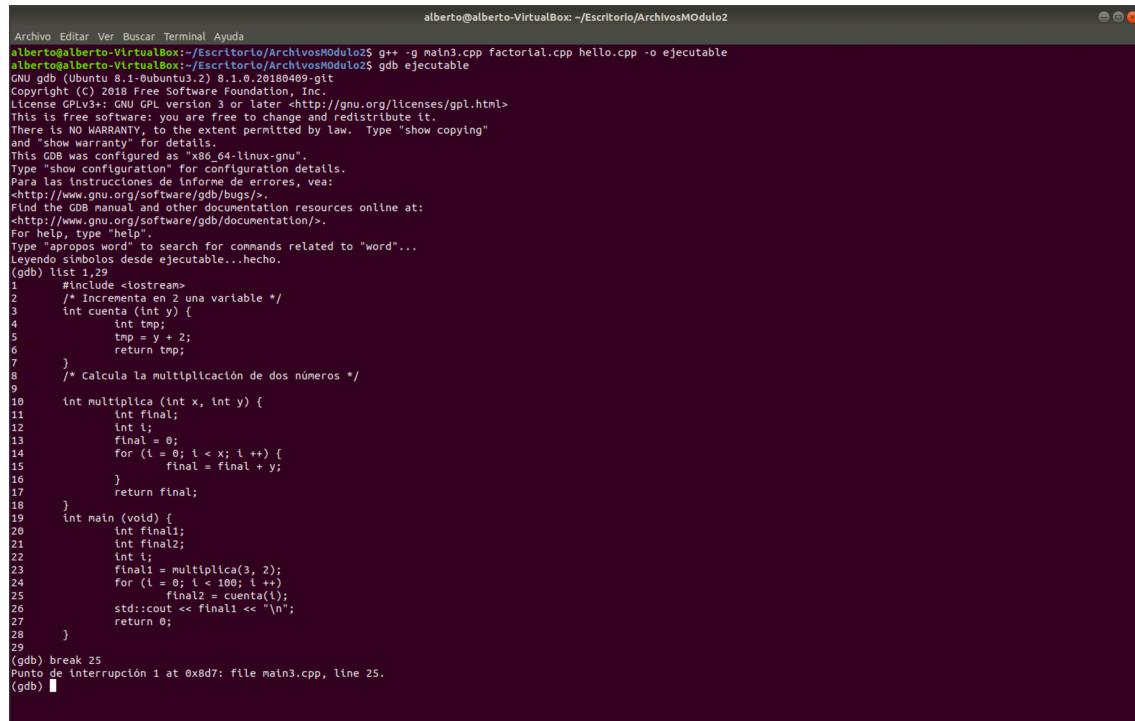
Como podemos observar en las imágenes 2.1 y 2.2, hemos utilizado el mismo comando que usamos en el ejercicio anterior para compilar el programa y ejecutar el depurador gdb. Una vez en el compilador, mediante la orden *break factorial*, vemos como se establece un punto de ruptura en la línea del main donde se encuentra la llamada a la función factorial. A continuación, ejecutamos la orden *run* para ejecutar el programa. Al haberse puesto este punto de interrupción, en la llamada a factorial el depurador comienza a llamar a la función factorial paso a paso, permitiéndonos depurar el programa y encontrar un error, en el caso de que hubiese. Para avanzar al siguiente paso, ejecutamos simplemente la orden *next*, *step* o *continue*. Finalmente, una vez obtenido el resultado, utilizamos la orden *quit* para salir del depurador.

Lista de comandos:

- *g++ -g main.cpp factorial.cpp hello.cpp –o ejecutable*
- *gdb ejecutable*
- *break factorial*
- *run*
- *step múltiples veces.*
- *quit*

**Ejercicio 3.** Compila el código del siguiente programa y haz todas las configuraciones necesarias con gdb para mostrar el valor de la variable final2 justo antes de que se le asigne ningún valor dentro del bucle for. Describe todos los pasos que has seguido e incluye una captura de pantalla de los mismos.

Veamos lo que se pide:



The screenshot shows a terminal window with the following content:

```
alberto@alberto-VirtualBox:~/Escritorio/ArchivosModulo2$ g++ -g main3.cpp factorial.cpp hello.cpp -o ejecutable
alberto@alberto-VirtualBox:~/Escritorio/ArchivosModulo2$ gdb ejecutable
GNU gdb (Ubuntu 8.1.0-0ubuntu3.2) 8.1.0.20180409-gdb
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 was created using build-id 64-bit Linux-gnubin.
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"...
Leyenda: símbolos desde ejecutable...hecho.
(gdb) list 1,25
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 /* Calcula la multiplicación de dos números */
9
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 int main (void) {
20     int final1;
21     int final2;
22     int i;
23     final1 = multiplica(3, 2);
24     for (i = 0; i < 100; i++)
25         final2 = cuenta(i);
26     std::cout << final1 << "\n";
27     return 0;
28 }
```

(gdb) break 25  
Punto de interrupción 1 at 0x8d7: file main3.cpp, line 25.  
(gdb) ■

(Imagen 3.1)

```

alberto@alberto-VirtualBox: ~/Escritorio/ArchivosModulo2
Archivo Editar Ver Buscar Terminal Ayuda
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/>.
<http://www.gnu.org/software/gdb/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 1,29
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     /* Calcula la multiplicación de dos números */
9
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        final1 = multiplica(3, 2);
25        for (i = 0; i < 100; i++)
26            final2 = cuenta(i);
27        std::cout << final1 << "\n";
28    }
29
(gdb) break 25
Punto de interrupción 1 at 0x8d7: file main3.cpp, line 25.
(gdb) run
Starting program: /home/alberto/Escritorio/ArchivosModulo2/ejecutable

Breakpoint 1, main () at main3.cpp:25
25             final2 = cuenta(i);
(gdb) print final2
$1 = 0
(gdb) 
```

(Imagen 3.2)

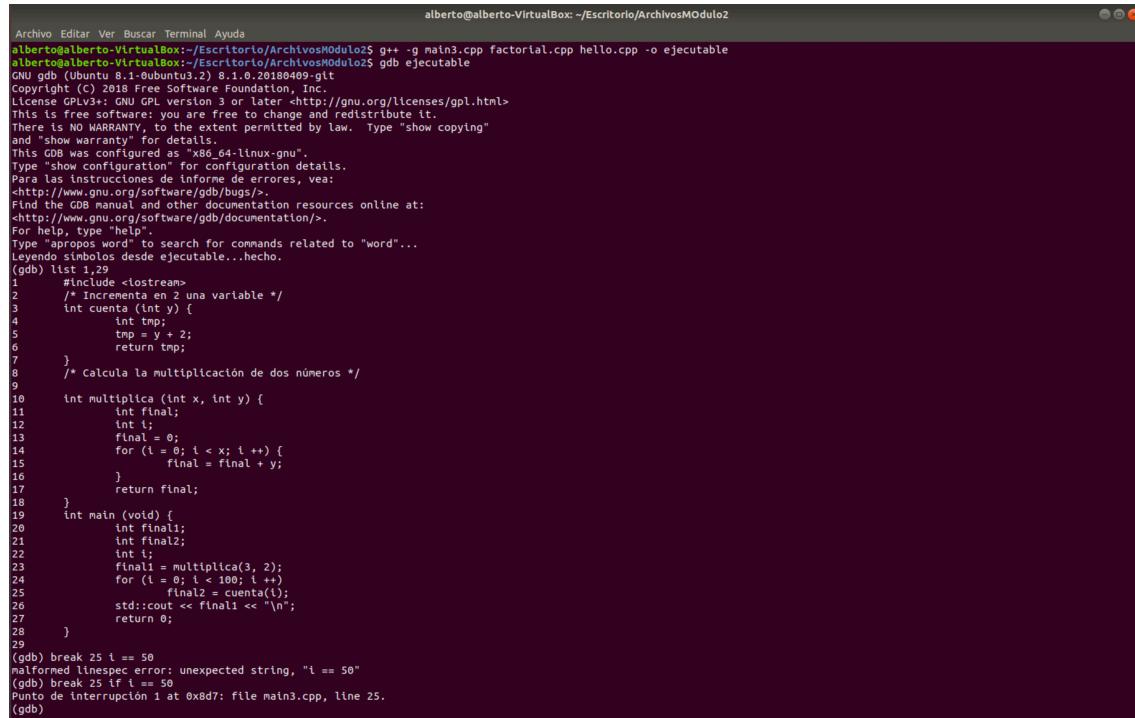
Fíjemonos en las imágenes 3.1 y 3.2. En este caso, tras la compilación del programa y ejecución del depurador, mostramos por pantalla el contenido del main para ver dónde realizar el punto de ruptura. Como podemos ver la variable final2 aparece en la línea 25, luego ponemos un punto de ruptura ahí como en el ejercicio anterior. A continuación, ejecutamos el programa y tras el punto de ruptura, mediante la orden *print final2*, mostramos por pantalla el valor de *final2*, sin que el programa finalice. Como vemos, todavía no se le ha asignado un valor en el *for*.

Lista de comandos:

- *g++ -g main3.cpp factorial.cpp hello.cpp –o ejecutable*
- *gdb ejecutable*
- *list 1,29*
- *break 25*
- *run*
- *print final2.*
- *quit*

**Ejercicio 4.** Siguiendo el ejercicio anterior, haz todas las configuraciones necesarias utilizando el depurador gdb para obtener el valor de la variable final2 cuando i vale 50. Muestra todos los comandos utilizados y el valor de las variables final2 e i.

Veamos la ejecución de lo pedido:



The screenshot shows a terminal window titled "alberto@alberto-VirtualBox: ~/Escritorio/ArchivosMODOulo2". The window displays a GDB session for a C++ program named "main3.cpp". The code in main3.cpp includes functions for summing and multiplying integers, and a main function that prints the results. A breakpoint is set at line 25 where i == 50. The terminal shows the command "break 25 i == 50" being entered, followed by the error message "malformed lINESPEC error: unexpected string, \"i == 50\"". The command "break 25 if i == 50" is then entered, and the message "Punto de interrupción 1 at 0x8d7: file main3.cpp, line 25." is displayed. The GDB prompt "(gdb)" is visible at the bottom.

```
Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMODOulo2$ g++ -g main3.cpp factorial.cpp hello.cpp -o ejecutable
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMODOulo2$ gdb ejecutable
GNU gdb (Ubuntu 8.1.0-20180409-1ubuntu1) 8.1.0.20180409-g1t
Copyright (C) 2018 Free Software Foundation, Inc.
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 symbols desde ejecutable...hecho.
(gdb) list 1,29
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  /* Calcula la multiplicación de dos números */
9
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 int main (void) {
20     int final;
21     int final2;
22     int i;
23     final1 = multiplica(3, 2);
24     for (i = 0; i < 100; i++) {
25         final2 = cuenta(i);
26         std::cout << final1 << "\n";
27     }
28 }
29
(gdb) break 25 i == 50
malformed lINESPEC error: unexpected string, "i == 50"
(gdb) break 25 if i == 50
Punto de interrupción 1 at 0x8d7: file main3.cpp, line 25.
(gdb)
```

(Imagen 4.1)

```

Archivo Editar Ver Buscar Terminal Ayuda
<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 1,29
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      /* Calcula la multiplicación de dos números */
9
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     int main (void) {
20         int final1;
21         int final2;
22         int i;
23         final1 = multiplica(3, 2);
24         for (i = 0; i < 100; i++)
25             final2 = cuenta(i);
26         std::cout << final1 << "\n";
27         return 0;
28     }
29
(gdb) break 25 if i==50
Punto de interrupción 1 en 0x8b7: file main3.cpp, línea 25.
(gdb) run
Arrancando programa: /home/alberto/Escritorio/ArchivosMÓdulo2/ejecutable

Breakpoint 1, main () en main3.cpp:25
25             final2 = cuenta(i);
(gdb) display i
1: i = 50
(gdb) display final2
2: final2 = 51
(gdb) quit
Una sesión de depuración está activa.

Inferior 1 [process 1916] will be killed.

; Salir de cualquier modo? (y o n) y
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMÓdulo2$ █

```

(Imagen 4.2)

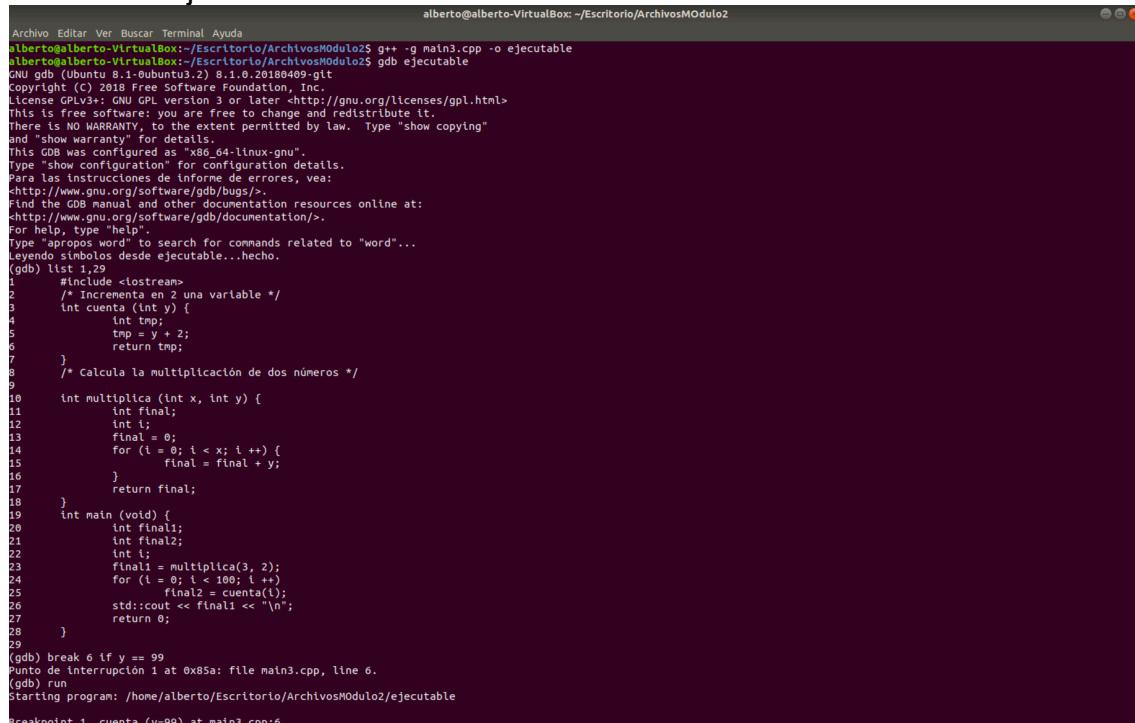
Como podemos observar en las imágenes 4.1 y 4.2, se añade una comprobación al punto de ruptura para que nos muestre el valor de *final2* cuando *i*=50, por lo que, al ejecutar el programa, podemos ver como al imprimir por pantalla el valor de *i*,(hemos usado en este caso la orden *display* para mostrar su equidad con *print*) vemos que *i*=50, y la variable *final2* = 51. Salimos con la orden *quit*.

Lista de comandos:

- *g++ -g main3.cpp factorial.cpp hello.cpp –o ejecutable*
- *gdb ejecutable*
- *list 1,29*
- *break 25 if i == 50*
- *run*
- *display i*
- *display final2.*
- *quit*

**Ejercicio 5.** Con el mismo código del ejercicio anterior no parece que la variable final2 pueda ser mayor de 101. Utilizando el depurador gdb haz, que sin tocar la variable final2, esta variable tenga un valor por encima de 1000 al final del programa. Describe todos los pasos que has seguido para conseguirlo e incluye la captura de pantalla correspondiente.

Veamos la ejecución:



The screenshot shows a terminal window titled "alberto@alberto-VirtualBox: ~/Escritorio/ArchivosMÓdulo2". The window contains a GDB session with the following commands and output:

```
Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMÓdulo2$ g++ -g main3.cpp -o ejecutable
alberto@alberto-VirtualBox:~/Escritorio/ArchivosMÓdulo2$ gdb ejecutable
GNU gdb (Ubuntu 8.1.0.20180409-1) 8.1.0.20180409-glt
Copyright (C) 2018 Free Software Foundation, Inc.
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..29
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 /* Calcula la multiplicación de dos números */
9
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 int main (void) {
20     int final;
21     int finalz;
22     int i;
23     final1 = multiplica(3, 2);
24     for (i = 0; i < 100; i++) {
25         finalz = cuenta(i);
26         std::cout << final1 << "\n";
27     }
28     return 0;
29 }
(gdb) break 6 if y == 99
Punto de interrupción 1 en 0x85a: file main3.cpp, línea 6.
(gdb) run
Starting program: /home/alberto/Escritorio/ArchivosMÓdulo2/ejecutable
Breakpoint 1, cuenta (y=99) at main3.cpp:6
```

(Imagen 5.1)

```

Archivo Editar Ver Buscar Terminal Ayuda
alberto@alberto-VirtualBox: ~/Escritorio/ArchivosModulo2
7     }
8     /* Calcula la multiplicación de dos números */
9
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    int main (void) {
20        int final1;
21        int final2;
22        int i;
23        final1 = multiplica(3, 2);
24        for (i = 0; i < 100; i++)
25            final2 = cuenta(i);
26        std::cout << final1 << "\n";
27        return 0;
28    }
(gdb) break 6 if y == 99
Punto de interrupción 1 en 0x87a: archivo main3.cpp, línea 6.
(gdb) run
Arrancando programa: /home/alberto/Escritorio/ArchivosModulo2/ejecutable
Breakpoint 1, cuenta (y=99) en main3.cpp:6
6          return tmp;
(gdb) set variable tmp = 1001
(gdb) step
7
(gdb) step
main () en main3.cpp:24
24          for (i = 0; i < 100; i++)
(gdb) step
25              std::cout << final1 << "\n";
(gdb) print final2
$1 = 1001
(gdb) step
6
27          return 0;
(gdb) step
28      }
(gdb) quit
Una sesión de depuración está activa.

Inferior 1 [proceso 4198] se va a matar.

; Salir de cualquier modo? (y o n) y
alberto@alberto-VirtualBox:~/Escritorio/ArchivosModulo2$ █

```

(Imagen 5.2)

Tras ejecutar los mismos comandos iniciales que en anteriores apartados y listar el main para poder ver dónde poner el punto de ruptura para realizar lo pedido, ponemos un break con comprobación para que se habilite sólo si se cumple la condición dicha. Para poder asignar el valor pedido a final2, tenemos que irnos a la última asignación a final2, es decir,  $i < 101 \Rightarrow 99$ . Luego una vez llegamos al punto de ruptura, podemos utilizar la orden *set* con sintaxis **set variable variable=valor**, cambiando el valor de final2 a 1001. Avanzando en la depuración con la orden *next* llegamos al final del programa y vemos cómo *final2=1001*. Finalmente, usamos la orden *quit* para salir del depurador.

Lista de comandos:

- *g++ -g main3.cpp -o ejecutable*
- *gdb ejecutable*
- *list 1,29*
- *break 6 if y == 99*
- *run*
- *set variable tmp = 1001*
- *step múltiples veces*
- *print final2*
- *quit*