Informe de Prácticas de Fundamentos de Bases de Datos

Práctica 2

Rodríguez Marrero, David
Larriba Moreno, Sergio

Índice

Introducción

La práctica se va a centrar en el funcionamiento del programa que usaremos para acceder y conectar con la base de datos. Como ya hemos practicado sentencias y programación en C todas esas partes van a ser bastantes sencillas y esta práctica se va a centrar en matrices, matrices que le llegan al programa que vamos a usar. Vamos a usar una librería ODBC que se basará en las primitivas para hablar con el programa usaremos las siguentes funciones: connect, disconnect, execute.

Lo bueno de estos odbcs es que si cambias de base de datos, con alguna actualización de odbc servirá para funcionar de nuevo perfectamente con la nueva base.

Han incluido en la práctica un zip con scripts para resolver y comprobar el resultado del menú que hemos implementado. Gracias a esos scripts podremos ver qué sentencias tenemos que imprimir en consola para un buen entendimiento del programa.

Nos dan un zip en el que hay un menú en c como modelo con su makefile Al final de la práctica, entregaremos todos los códigos de las consultas y el Makefile que hemos utilizado prácticamente igual al dado en Moodle. También entregaremos este informe en el cual hemos redactado nuestros pensamientos durante toda la práctica.

Stock

Stock: aquí se solicita un productcode por consola y se devuelve el quantityinstock de ese producto según nuestra base de datos. Lo que hemos hecho es una sentencia simple de SQL que pedía el productcode y devolvía el quantityinstock y hemos usado la función de BindParameter para registrar el productcode y el BindCoL para escribir el resultado. Esas funciones están implementadas en products.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.

```
eps@vmlabsdocentes:~/Escritorio/Practica2$ ./products_stock.sh
spawn ./menu
Este programa mostrará un menú de 4 opciones
Segun el numero que insertes accederas al submenú correspondiente

 (1) Products
 (2) Orders
 (3) Customers
 (4) Exit

Enter a number that corresponds to your choice > 1

 (1) Stock
 (2) Find
 (3) Back

Enter a number that corresponds to your choice > 1

Enter productcode > S10_1678

Unidades en stock: 7933
```

Find

Find: lee una cadena de productname e imprime todos los productcodes en los que el productname contenga esa cadena. Lo que hemos hecho es una sentencia simple de SQL que pedía una cadena de productname y devolvía el productcode, a esa cadena posteriormente le hemos concatenado '%' para que lo registre como una cadena y hemos usado la función de BindParameter para registrar el productname y el BindCoL para escribir el resultado. Esas funciones están implementadas en products.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.

```
Este programa mostrará un menú de 4 opciones
Segun el numero que insertes accederas al submenú correspondiente

 (1) Products
 (2) Orders
 (3) Customers
 (4) Exit

Enter a number that corresponds to your choice > 1

 (1) Stock
 (2) Find
 (3) Back

Enter a number that corresponds to your choice > 2

Enter productname > Harley

S10_1678 Harley

S10_4698 Harley

S18_2625 Harley
```

Open

Open: imprime una lista de los pedidos que no se hayan enviado ordenados por ordernumber. Lo que hemos implementado es una sentencia simple de SQL que no pide ningún parámetro y que imprime todos los orders que cumplen el shippeddate nulo y lo imprimimos con BindCol. Esas funciones están implementadas en orders.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.

```
Este programa mostrará un menú de 4 opciones
Segun el numero que insertes accederas al submenú correspondiente

 (1) Products
 (2) Orders
 (3) Customers
 (4) Exit

Enter a number that corresponds to your choice > 2

 (1) Open
 (2) Range
 (3) Detail
 (4) Back

Enter a number that corresponds to your choice > 1

10167

10248

10260

10262

10334

10401

10407

10414

10420

10421

10422

10423

10424

10425
--------------------------------OK
```

Range

Range: solicitamos dos fechas en formato y devuelve los pedidos solicitados entre esas fechas ordenados por ordernumber. Lo que hemos hecho es una sentencia en SQL que pide un intervalo de fechas, después, mediante un bucle, separo los datos en dos fechas distintas y leo las dos con dos BindParameter. Por último, utilizamos BindCoL para imprimir los atributos que queramos. Esas funciones están implementadas en orders.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.

```
Este programa mostrará un menú de 4 opciones
Segun el numero que insertes accederas al submenú correspondiente

 (1) Products
 (2) Orders
 (3) Customers
 (4) Exit

Enter a number that corresponds to your choice > 2

 (1) Open
 (2) Range
 (3) Detail
 (4) Back

Enter a number that corresponds to your choice > 2

Enter dates (YYYY-MM-DD - YYYY-MM-DD) > 2003-01-10 - 2003-04-21
10102 2003-01-10 2003-01-14

10103 2003-01-29 2003-02-02

10104 2003-01-31 2003-02-01

10105 2003-02-11 2003-02-12

10106 2003-02-17 2003-02-21

10107 2003-02-24 2003-02-26

10108 2003-03-03 2003-03-08

10109 2003-03-10 2003-03-11

10110 2003-03-18 2003-03-20

10111 2003-03-25 2003-03-30

10112 2003-03-24 2003-03-29

10113 2003-03-26 2003-03-27

10114 2003-04-01 2003-04-02

10115 2003-04-04 2003-04-07

10116 2003-04-11 2003-04-13
```

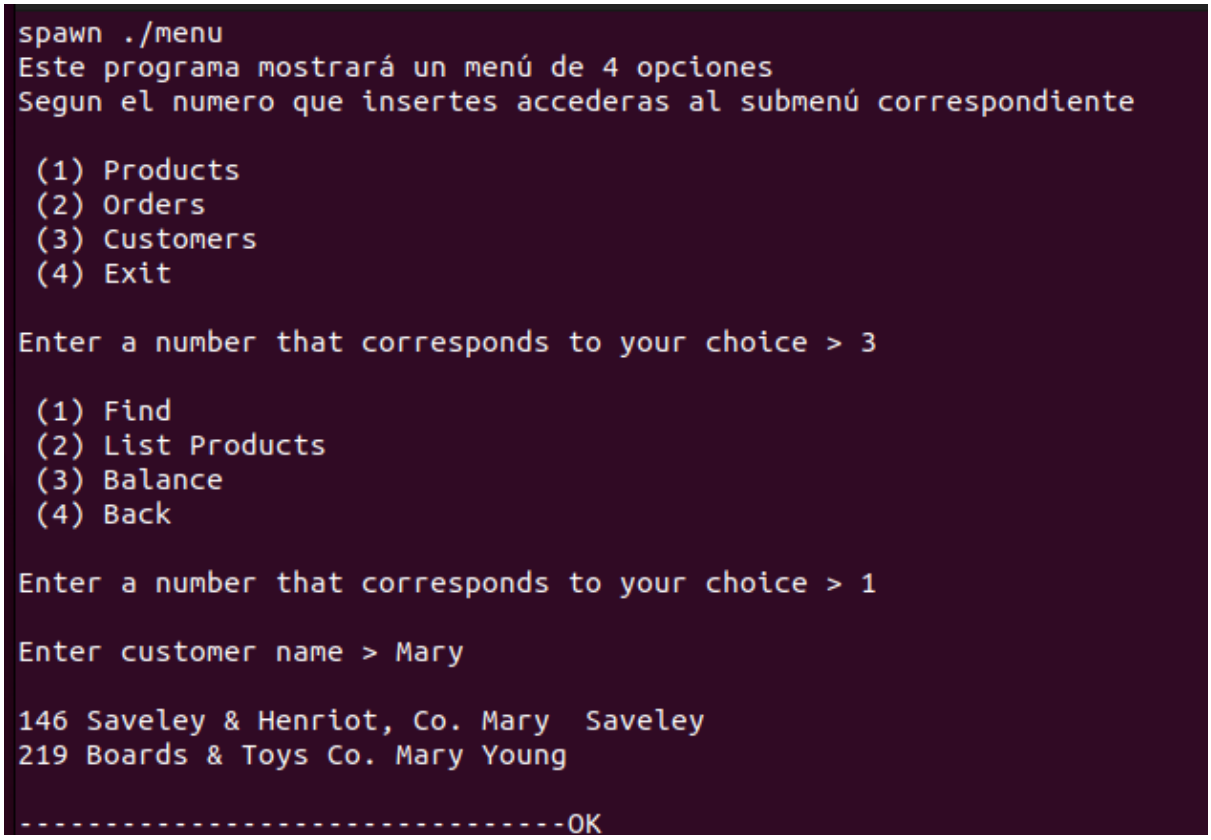
Detail

Detail: solicitamos un order y devuelve todos los detalles de ese pedido ordenado por orderlinenumber. Lo que hemos hecho es una sentencia de SQL que joinea orders con ordersdetails para obtener todos los detalles del pedido y hemos leído un ordernumber del cual para la cantidad total del pedido hemos tenido que multiplicar las unidades por el precio de cada una y guardarlo en una variable renombrada. El ordernumber lo hemos leído gracias a BindParameter y con un formato pedido en la práctica y en los scripts hemos impreso las variables guardadas por BindCol con printf escribir el resultado. Esas funciones están implementadas en orders.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.

```
 (1) Products
 (2) Orders
 (3) Customers
 (4) Exit

Enter a number that corresponds to your choice > 2

 (1) Open
 (2) Range
 (3) Detail
 (4) Back

Enter a number that corresponds to your choice > 3

Enter ordernumber > 10100
----------------------
2003-01-06 Shipped
--------------------
Totalcost: 10223.83
--------------------
S24_3969 49 35.29
S18_2248 50 55.09
S18_1749 30 136.00
S18_4409 22 75.46
```

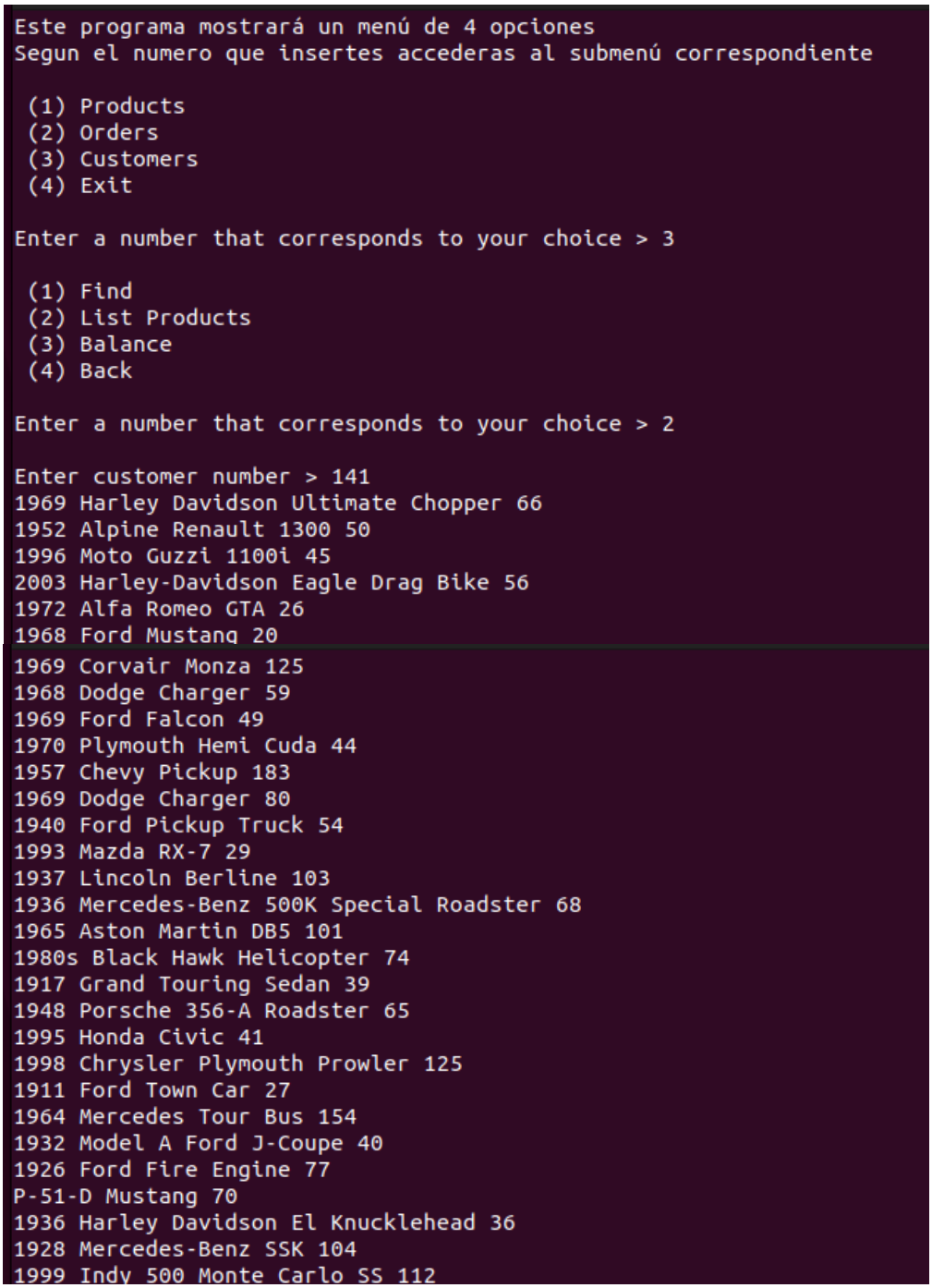
Find

Find: se pide un contactname y se imprimen todos los clientes que tengan esa cadena como nombre o apellido con todos sus detalles. Lo que hemos ejecutado es una sentencia de SQL que pedía un nombre y a esa cadena posteriormente le hemos concatenado '%%' para que lo registre como una cadena y hemos usado dos funciones de BindParameter para registrar el nombre como firstname o como lastname y el BindCoL para escribir todas las columnas con el número de cliente, el nombre del cliente y su nombre completo de contacto. Esas funciones están implementadas en customers.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.



List Products

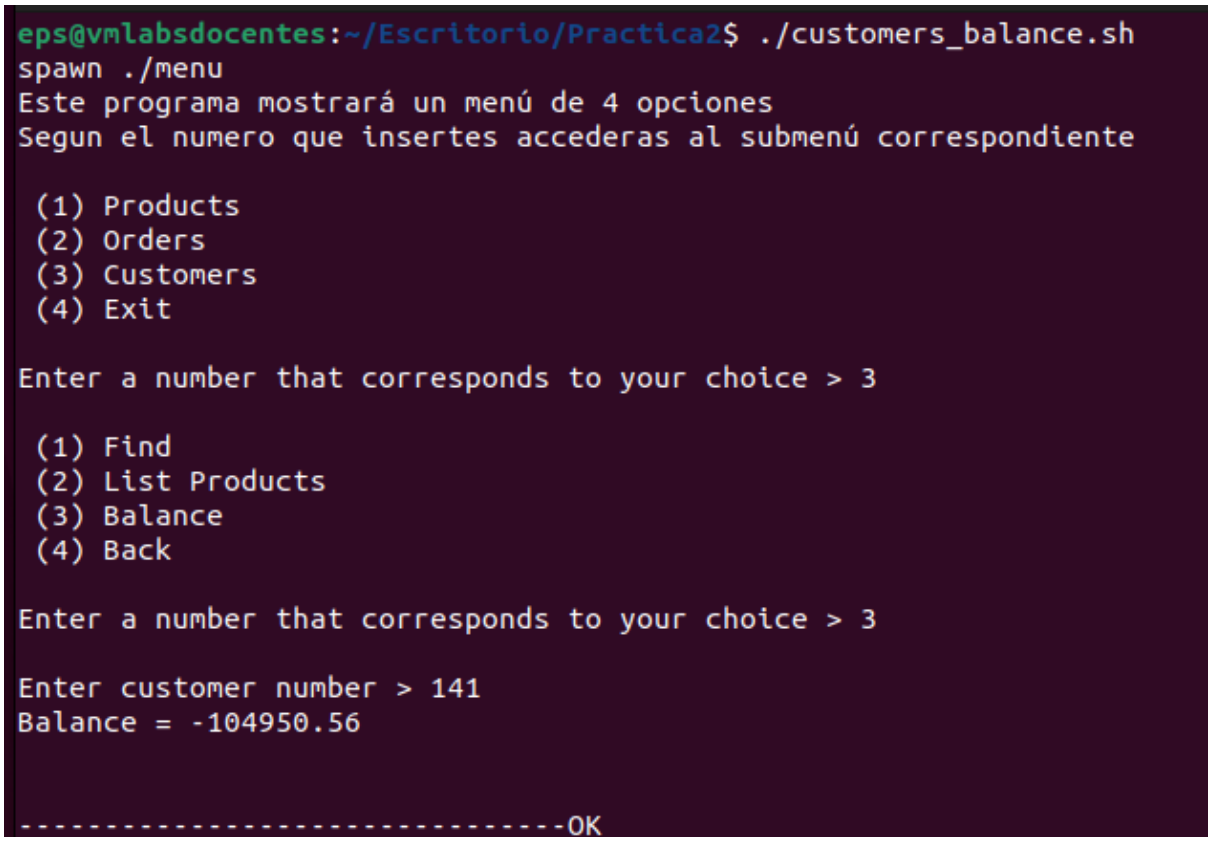
List Products: se solicita un numero de cliente y se imprimen todos los pedidos solicitados por ese cliente con su nombre y la cantidad de unidades solicitadas. Lo que hemos hecho es una sentencia de SQL que pedía un customernumber y devolvía una suma de la cantidad de cada producto en un order y su nombre. El numero lo hemos registrado con la función de BindParameter y hemos usado el BindCol para registrar los valores que vamos a imprimir. Esas funciones están implementadas en customers.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.



```
Este programa mostrará un menú de 4 opciones
Segun el numero que insertes accederas al submenú correspondiente

 (1) Products
 (2) Orders
 (3) Customers
 (4) Exit

Enter a number that corresponds to your choice > 3

 (1) Find
 (2) List Products
 (3) Balance
 (4) Back

Enter a number that corresponds to your choice > 2

Enter customer number > 141
1969 Harley Davidson Ultimate Chopper 66
1952 Alpine Renault 1300 50
1996 Moto Guzzi 1100i 45
2003 Harley-Davidson Eagle Drag Bike 56
1972 Alfa Romeo GTA 26
1968 Ford Mustang 20
1969 Corvair Monza 125
1968 Dodge Charger 59
1969 Ford Falcon 49
1970 Plymouth Hemi Cuda 44
1957 Chevy Pickup 183
1969 Dodge Charger 80
1940 Ford Pickup Truck 54
1993 Mazda RX-7 29
1937 Lincoln Berline 103
1936 Mercedes-Benz 500K Special Roadster 68
1965 Aston Martin DB5 101
1980s Black Hawk Helicopter 74
1917 Grand Touring Sedan 39
1948 Porsche 356-A Roadster 65
1995 Honda Civic 41
1998 Chrysler Plymouth Prowler 125
1911 Ford Town Car 27
1964 Mercedes Tour Bus 154
1932 Model A Ford J-Coupe 40
1926 Ford Fire Engine 77
P-51-D Mustang 70
1936 Harley Davidson El Knucklehead 36
1928 Mercedes-Benz SSK 104
1999 Indy 500 Monte Carlo SS 112
```

Balance

Balance: se solicita un numero de cliente y se devuelve el saldo actual del mismo cogiendo los pagos y restando el valor de todos los productos comprados. Lo que hemos implementado es una sentencia de SQL complicada en la que hacemos selects anidados para obtener la resta de la cantidad de dinero amount del cliente y lo que ha comprado registrado con su identificador. Esto lo podemos expresar en una sola sentencia pero sin embargo necesitamos dos BindParameter ya que en las dos consultas anidadas necesitamos el identificador del cliente. Finalmente, registramos el valor final de la resta en un BindCol y lo imprimimos. Esas funciones están implementadas en customers.c y las llamamos desde el menú. Aquí tenemos el resultado del script correspondiente.



Splint

En este apartado tuvimos que ejecutar un Split para detectar errores en nuestro código relacionados con el valor nulo.

Inicialmente, al ejecutar el splint, nos salían los siguientes errores:

```
menu.c:28:13: Global env initialized to null value: env = NULL
  A reference with no null annotation is assigned or initialized to NULL. Use
  /*@null@*/ to declare the reference as a possibly null pointer. (Use
  -nullassign to inhibit warning)
menu.c:28:13: Global env initialized to null value: SQLHENV env = NULL = NULL
menu.c:29:13: Global dbc initialized to null value: dbc = NULL
menu.c:29:13: Global dbc initialized to null value: SQLHDBC dbc = NULL = NULL
menu.c:30:15: Global stmt initialized to null value: stmt = NULL
menu.c:30:15: Global stmt initialized to null value:
                SQLHSTMT stmt = NULL = NULL
menu.c: (in function main)
menu.c:40:17: Return value (type int) ignored: ShowProductMenu()
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalint to inhibit warning)
menu.c:48:17: Return value (type int) ignored: ShowCustomersMenu()
menu.c: (in function ShowProductMenu)
menu.c:103:17: Return value (type int) ignored: StockMenu()
menu.c:109:17: Return value (type int) ignored: FindMenu()
menu.c: (in function ShowOrdersMenu)
menu.c:208:17: Return value (type int) ignored: OpenMenu()
menu.c:214:17: Return value (type int) ignored: RangeMenu()
menu.c:220:17: Return value (type int) ignored: DetailMenu()
menu.c: (in function ShowCustomersMenu)
menu.c:355:17: Return value (type int) ignored: FindMenu2()
menu.c:361:17: Return value (type int) ignored: ListProductsMenu()
menu.c:367:17: Return value (type int) ignored: BalanceMenu()
odbc.c: (in function odbc_extract_error)
odbc.c:31:47: Format argument 3 to printf (%d) expects int gets SQLINTEGER:
                native
  Type of parameter is not consistent with corresponding code in format string.
  (Use -formattype to inhibit warning)
   odbc.c:31:28: Corresponding format code
orders.c: (in function rRangeMenu)
orders.c:67:25: Index of null pointer fecha: fecha
  A possibly null pointer is dereferenced. Value is either the result of a
  function which may return null (in which case, code should check it is not
  null), or a global, parameter or structure field declared with the null
  qualifier. (Use -nullderef to inhibit warning)
menu.c:28:9: Variable exported but not used outside menu: env
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
menu.c:29:9: Variable exported but not used outside menu: dbc
menu.c:30:10: Variable exported but not used outside menu: stmt
```

Sin embargo, después de revisar los archivos uno a uno gracias a los fallos perfectamente indicados por el comando, hemos podido reducir ese número de fallos a 0 por lo que hemos adjuntado a los archivos de la entrega el archivo resultante splint.log que está vacío debido a la ausencia de errores. Después de ver la capacidad de este comando nos hemos dado cuenta de que aunque parezca que todos los scripts van bien y no hay ningún fallo de sintaxis según VSCode, puede haber otros fallos muy importantes al tratar con bases de datos que son los NULLS de los cuales en c solemos abusar mucho en nuestro código.

Conclusiones

Las conclusiones que hemos sacado de esta práctica son las siguientes:

- La librería odbc nos permite detallar una conexión con una base de datos en c, lenguaje en el que no se suele hacer consultas a una base de datos. Esto implica una apertura en los lenguajes de programación que podríamos usar al pensar en una aplicación con conexión a una base de datos.

- El conocimiento de SQL nos permite hacer sentencias para obtener los resultados impresos en consola y así hemos podido utilizar el conocimiento adquirido en la práctica 1. Por otra parte, hemos podido ver la facilidad con la que podemos realizar otros programas sólo sabiendo usar SQL y aprendiendo algunas funciones en el lenguaje deseado.

- El conocimiento de C nos ha permitido entender las funciones de la librería ODBC y poder trabajar con ellas como funciones normales de C. También hemos aprendido a estructurar mejor nuestro código utilizando funciones y distintos documentos en los que podemos entender mejor el funcionamiento de la aplicación

En conclusión, hemos aprendido a hacer aplicaciones que usen una base de datos adecuada y arbitraria en la que podamos poner cualquier tipo de datos desde C que es un lenguaje en el que no se suelen hacer este tipo de aplicaciones. Esta actividad nos da una libertad monumental para poder crear una aplicación cualquiera y conectarla a donde deseemos mediante este método.

Finalmente, hemos visto una de las utilidades de lo aprendido en clases teóricas y prácticas con una actividad que nos puede servir en un futuro y nos permite desarrollar nuestra creatividad para tener ganas de seguir aprendiendo sobre las bases de datos y su importancia en el nuevo mundo tecnológico.