Memoria de práctica:
VehiCom

Autora: Serena Lombardi
Y5282149Y

Curso 2021/22

Asignatura de Fundamentos de Programación
Grado en Ingeniería Informática
UNED
Centro asociado de Málaga

Estructura del proyecto

```
1  /*****************************************************
2  * NOMBRE: #Serena#
3  * PRIMER APELLIDO: #Lombardi#
4  * SEGUNDO APELLIDO: #---#
5  * DNI: #Y5282149Y#
6  * EMAIL: #slombardi6@alumno.uned.es#
7  *****************************************************
8  El programa VehiCom gestiona el alquiler de hasta 20
9  vehiculos para 50 clientes máximo
10 *****************************************************/
11
12 #include <stdio.h>
13 #include <math.h>
14 #include "print_to_console.h"
15 #include "data_mgmt.h"
16 #include "calendar.h"
17
18 /*=====================================================
19 Programa principal
20 =====================================================*/
21
22 char option;
23
24 int main (){
25   InitCustomerTable();
26   InitVehicleTable();
27   InitActVehicleTable();
28   InitCStatsTable();
29   InitMonthsArray();
30   FirstScreen();
31   return 0;
32   }
```

El proyecto consta de varios módulos.

main_program es el módulo que controla la inicialización de las tablas y la ejecución del programa y contiene el int main().

print_to_console incluye los subprogramas relacionados con lo que se muestra en consola.

data_mgmt se encarga de crear, inicializar y manejar tipos de datos y tablas.

calendar reutiliza parte del código de la tercera práctica, según pide el enunciado.

En el proyecto no hay de momento control de errores a nivel de los datos introducidos por el usuario: se confía en que el usuario introduzca datos con tipo y rango de valores compatibles con las instrucciones. Se importan los módulos **math.h** y **string.h**.

Primera pantalla

Desde **main_program** se usa el método **FirstScreen()** de **print_to_console,** guardando en una variable tipo **char** el resultado, o sea la letra que el usuario elige para decidir a qué pantalla del programa entrar.
FirstScreen() imprime el texto de la primera pantalla usando el subprograma **PrintLine()**, que permite imprimir la línea manejando los espacios con 2 variables (una para los espacios

```
Vehicom: Gestion de vehiculos compartidos
    Alta nuevo cliente      (Pulsar C)
    Alta nuevo vehiculo     (Pulsar V)
    Activar vehiculo        (Pulsar A)
    Devolver vehiculo       (Pulsar D)
    Resumen mensual         (Pulsar R)
    Salir                   (Pulsar S)
Teclear una opcion valida (C|V|A|D|R|S)?
```

al principio, una para el total de caracteres de la línea).

FirstScreen() luego pide al usuario que seleccione una letra para seguir a otra pantalla o salir del programa (C|V|A|D|R|S). Se usa un switch para controlar qué subprograma se usará a continuación. Si el usuario introduce un carácter no aceptado, se entenderá que desea salir del programa: se ejecutará la opción default del switch.

Alta nuevo cliente

```
Teclear una opcion valida (C|V|A|D|R|S)?
C

Alta nuevo cliente:

    Identificador (numero entre 1 y 50)?5
    Nombre (entre 1 y 10 caracteres)?Pablo
    Apellido (entre 1 y 20 caracteres)?Perez

Datos correctos (S/N)?S
Otro cliente (S/N)?S

Alta nuevo cliente:

    Identificador (numero entre 1 y 50)?21
    Nombre (entre 1 y 10 caracteres)?Pedro
    Apellido (entre 1 y 20 caracteres)?Lopez

Datos correctos (S/N)?S
Otro cliente (S/N)?N


Vehicom: Gestion de vehiculos compartidos
```

La opción de alta de nuevo cliente se maneja desde el subprograma **AddNewCustomer()**. Este usa la función **NewCustomerText()** para mostrar el texto de la nueva pantalla, pedir los datos al usuario y guardarlos solo si el usuario lo confirma tecleando 'S'.

Si el usuario no teclea ni 'S' ni 'N', se considera como si hubiera tecleado 'N'. **NewCustomerText()** devuelve el carácter tecleado como respuesta a "Otro cliente (S/N)?", que es luego usado por el procedimiento **RepeatNewCustomer()** para valorar si procede volver a ejecutar **NewCustomerText()** para un nuevo usuario o para sobreescribir un usuario anteriormente grabado.

Cuando el usuario termine de grabar usuarios, **AddNewCustomer()** vuelve a ejecutar **FirstScreen(),** o sea la pantalla principal.

La información de cada cliente se almacena en las variables de un struct llamado **customer**; se ha creado una tabla de 50 registros para almacenar la información de todos los registros. La tabla se inicializa con **InitCustomerTable()**.

Alta nuevo vehículo

La estructura de esta parte del programa es muy parecida a la sección anterior. Se ha creado un struct **vehicle**, una tabla de 20 registros tipo **vehicle** y los siguientes subprogramas: **NewVehicleText()**, **RepeatNewVehicle()**, **AddNewVehicle()**. La tabla se inicializa con **InitVehicleTable()**.

```
Teclear una opcion valida (C|V|A|D|R|S)?
V

Alta nuevo vehiculo:

    Identificador (numero entre 1 y 20)?3
    Tipo de vehiculo (C/B/P)?C
    Descripcion (entre 1 y 20 caracteres)?Seat Leon Blanco
    Radio (hasta 1000 metros)?5423
    Angulo (entre 0.00 y 360.00 grados)?82.08

Datos correctos (S/N)?S
Otro vehiculo (S/N)?S

Alta nuevo vehiculo:

    Identificador (numero entre 1 y 20)?7
    Tipo de vehiculo (C/B/P)?B
    Descripcion (entre 1 y 20 caracteres)?NH Electrica Roja
    Radio (hasta 1000 metros)?8876
    Angulo (entre 0.00 y 360.00 grados)?244.08

Datos correctos (S/N)?S
Otro vehiculo (S/N)?N


Vehicom: Gestion de vehiculos compartidos
```

Activación de vehículo

El subprograma completo de activación de vehículos es **ActivateVehicle()**, que controla también la nueva ejecución de **FirstScreen()** al terminar.

Este incluye varios otros subprogramas, incluido en el módulo **print_to_console** si tratan de imprimir texto o **data_mgmt** para el manejo de las tablas.

ActivateVehicleText() crea el texto y llama a los subprogramas de manejo de datos. **AvailableVehicles()** busca un vehículo que cumpla los criterios insertados por el usuario averiguando que no se haya incluido ya en la tabla donde se guardan los 5 resultados para mostrar a pantalla. Se calcula también la distancia entre usuario y vehículo en coordenadas polares con **PolarCoordDistance()**.

```
155   /*calculo de distancias en coordenadas polares*/
156   float PolarCoordDistance(float rho1, float rho2, float theta1, float theta2) {
157     float result;
158     theta1 = DegreesToRadians(theta1);
159     theta2 = DegreesToRadians(theta2);
160     result = pow(rho1, 2) + pow(rho2, 2) - 2*rho1*rho2*cos(theta1-theta2);
161     if (result >= 0) {
162       return sqrt(result);
163       } else {
164         return sqrt(-result);
165         }
166     }
```

Si no se encuentran más vehículos compatibles, se muestran a pantalla registros con letra N y valores nulos. En este ejemplo, solo hay una bicicleta disponible en toda la tabla de vehículos:

```
Teclear una opcion valida (C|V|A|D|R|S)?
A

Activar vehiculo:

    Identificador cliente?5
    Ubicacion cliente: Radio?1805.5
    Ubicacion cliente: Angulo?45
    Tipo de vehiculo (C/B/P/T)?B
    Fecha de activacion (DD/MM/AA)?12/01/22
    Hora de activacion (HH:MM)?12:45

            Vehiculos disponibles:

    Ref.                Identificador       Tipo                Distancia           Rumbo
    1                    6                  B                   2164.95             130.00
    2                    0                  N                      0.00             0.00
    3                    0                  N                      0.00             0.00
    4                    0                  N                      0.00             0.00
    5                    0                  N                      0.00             0.00
```

Tras elegir uno de los vehículos, se muestran los datos del vehículo elegido. Si se elige un vehículo no válido, aparecerá un mensaje de error en lugar del tipo, para que el usuario pueda evitar confirmar su elección.

```
    Ref. vehiculo seleccionado?5

            Datos del vehiculo seleccionado:

    Identificador: 0        Tipo: No se ha seleccionado un vehiculo valido
    Descripcion:
    Distancia desde cliente:    0.00 metros
    Rumbo desde cliente: 0.00 grados
```

Si el usuario teclea S, se guardarán los datos y se confirmará en pantalla que se ha activado el vehículo. Si el usuario teclea otro carácter, volverá a ver la pantalla principal.

Al guardar los datos, se transfieren los datos del vehículo a la tabla de vehículos activados (**avTable**, cuyos registros tienen una variable más para el id del cliente que ha activado el vehículo) y se borra de la primera tabla de vehículos para indicar que no está disponible. No

se pregunta al usuario si quiere activar un nuevo vehículo, porque no aparece la opción en el enunciado.
A continuación se muestra un ejemplo de ejecución del programa.

```
Teclear una opcion valida (C|V|A|D|R|S)?
A

Activar vehiculo:

    Identificador cliente?3
    Ubicacion cliente: Radio?7654
    Ubicacion cliente: Angulo?64.08
    Tipo de vehiculo (C/B/P/T)?T
    Fecha de activacion (DD/MM/AA)?20/10/21
    Hora de activacion (HH:MM)?12:28

            Vehiculos disponibles:

    Ref.                Identificador       Tipo                Distancia           Rumbo
    1                    3                  C                   7157.12             7.92
    2                   11                  C                   7164.93             15.92
    3                   15                  C                   7172.89             19.92
    4                   19                  C                   7524.93             -21.08
    5                   10                  P                   7542.22             -30.08

    Ref. vehiculo seleccionado?3

            Datos del vehiculo seleccionado:

    Identificador: 15       Tipo: Coche
    Descripcion: vehiculo de color azul
    Distancia desde cliente: 7172.89 metros
    Rumbo desde cliente: 19.92 grados

    Datos correctos (S/N)?S

    El vehiculo con identificador: 15, ha sido ACTIVADO desde las 12:28 horas del dia 20/10/21.
```

Devolución de vehículo

La estructura de este apartado es coherente con la estructura de los apartados previos: **ReturnVehicle()** y **ReturnVehicleText()** en **print_to_console**, **SaveReturnData()** en **data_mgmt**.
Al confirmar con 'S' que se desea devolver el vehículo, se guarda de nuevo la información del vehículo en la primera tabla **vTable** y se borra de la tabla de los vehículos activados. Se guardan los datos para el resumen del siguiente apartado. Se ha decidido no desarrollar control de errores a nivel de los datos introducidos por el usuario, confiando en que el usuario no confirme los datos si no son correctos.
A continuación se muestra un ejemplo de ejecución.

```
Teclear una opcion valida (C|V|A|D|R|S)?
D

Devolver vehiculo:

    Identificacion vehiculo?15
    Nueva ubicacion vehiculo: radio?6211
    Nueva ubicacion vehiculo: angulo?179.64
    Tiempo de uso (en minutos)?108

            Datos del vehiculo:

    Identificador: 15          Tipo: Coche
    Descripcion: vehiculo de color azul
    Nueva ubicacion: radio = 6211.00 metros
    Nueva ubicacion: angulo = 179.64 grados


    Datos correctos (S/N)?S

    El vehiculo con identificador: 15, ha sido DEVUELTO despues de haber sido utilizado durante 108 minutos.
```

Resumen mensual uso de cliente

En el resumen se ha decidido contabilizar los minutos en el mes de activación del vehículo, sin tener en cuenta si, debido a la fecha y al número de minutos, el coche es devuelto ya durante el mes siguiente. La estructura del programa es coherente con los apartados anteriores, incluyendo “MonthlyStats” en los nombres de subprogramas.

```
Teclear una opcion valida (C|V|A|D|R|S)?
R

Resumen mensual de cliente
Identificador cliente?5
Seleccion Mes?10
Seleccion Anno (aaaa)?2021

            Resumen uso cliente: Pablo Perez

            Octubre                 2021
            ____________________________
             L   M   M   J   V |  S   D
                               |
                             1 |  2   C
             B   5   6  BC   8 |  9  10
            TO  12  13   P  BP | 16   C
            18  19   C  21  22 | 23  24
            25  26  27  28  29 |  P   C


Tiempo de uso Bicicletas: 334 minutos
Tiempo de uso Patinetes: 225 minutos
Tiempo de uso Coches: 503 minutos

B: Bicicleta P: Patinete C: Coche TO: Todos

Mostrar otro mes (S/N)?S

Resumen mensual de cliente
```

Para realizar el recuento de minutos por vehículo, se ha realizado una nueva tabla para los registros de uso de los coches, con el id del cliente, los minutos, mes y año de activación del coche y tipo de vehículo. Se pueden grabar un máximo de 99 registros. Se aceptan fechas desde el año 2000 hasta el año 2099: en este apartado los años se introducen con 4 dígitos, no con 2 como en anteriores apartados, porque así aparece en el enunciado.