

Asignatura: **Fundamentos de Programación**

Práctica 5: **Programación con C+/- de un sistema gestor de la distribución de fármacos**

Alumno: **rsanchez628@alumno.uned.es**

1.- Descripción de los módulos.

Paquete Almacén

Paquete Almacén



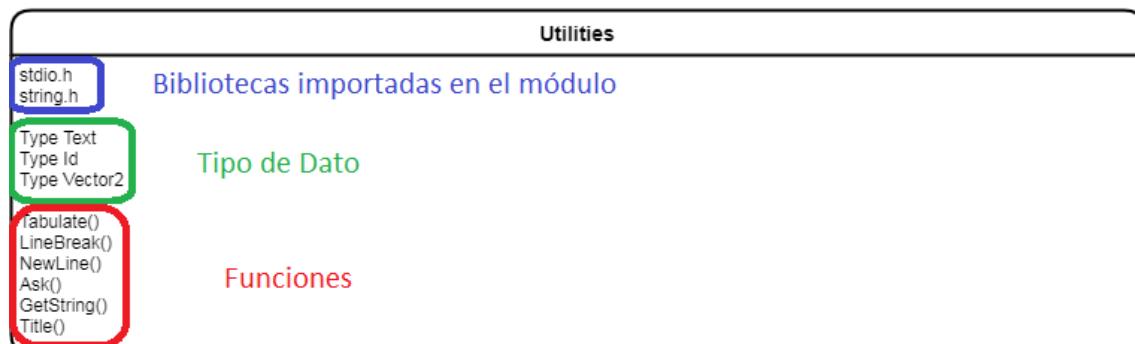
Módulo servicios

```
Utilities.h [Gestor_FarmaDron] - Code::Blocks 8.02
Archivo Edición Ver Buscar Proyecto Generar Depurar Herramientas Prácticas Complementos Configuración Ayuda
C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Utilities.h Comilar target: Release
Utilities.h x Utilities.cpp
1 #pragma once
2 #include <stdio.h>
3 #include <string.h>
4
5 typedef char Text[100];
6
7 typedef Text Id[2];
8
9 typedef int Vector2[2];
10
11 void Tabulate(int space);
12
13 void LineBreak();
14
15 void NewLine();
16
17 int Ask(Text question); //Y->1; N->0
18
19 void GetString(Text question, Text &answer);
20
21 void Title(Text title);
```

C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Utilities.h WINDOWS-1252 Línea 21, Columna 24 Insertar Lectura/Escritura ccuned

Elementos del módulo

Módulo Servicios



- **Tabulate**. Escribiendo por pantalla la función dejará el número de espacios en blanco que se le pase por parámetro.
- **LineBreak**. Escribiendo por pantalla, Salto de línea.
- **NewLine**. Salto de línea haciendo un sangrado de 5 espacios en blanco al empezar el nuevo párrafo.
- **Ask**. Escribe por pantalla la pregunta que se le pasa por parámetro. Devuelve un “1” si la respuesta del usuario es un *sí* y un “0” si el usuario responde *no*. Discrimina las respuestas que no se corresponden a las opciones *sí/no*.

```

Ref. Paciente <entre 1 y 20> 1
Número de envíos? 2
Número de días entre cada envío <Entre 1 y 15 días>? 17
El sistema no procesa 17 días entre envíos
Número de días entre cada envío <Entre 1 y 15 días>? 2
Día del primer envío? 1
Mes del primer envío? 1
Año del primer envío? 2021
Nombre del fármaco <Entre 1 y 20 caracteres>? Aspirina
Peso fármaco <Menor de 3000 gramos>? 100
Unidades de fármaco? 3
Otro fármaco <S/N>? b
Introdujo: b
Otro fármaco <S/N>? _

```

- **GetString.** Escribe por pantalla la pregunta que se le pasa por parámetro en la variable *question*. Lee la cadena de texto que el usuario escribe por pantalla, respetando los espacios en blanco y almacena en la variable *answer* la cadena de texto que escribió el usuario.
- **Title.** El texto que se le pasa por parámetro aparecerá por pantalla con un determinado formato.

1	28/10/2021	Colutorio	300	1
3	28/10/2021	Jarabe	100	3
4	28/10/2021	Panales	500	1

```

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen             <Pulsar M>
    Alta paciente almacen   <Pulsar A>
    Ubicar pacientes         <Pulsar U>
    Nuevo pedido              <Pulsar N>
    Lista diaria de pedidos  <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I|M|A|U|N|L|P|R|S>?
A

Alta nuevo paciente:      Texto con formato Title
Codigo almacen? <entre 1-10> _

```

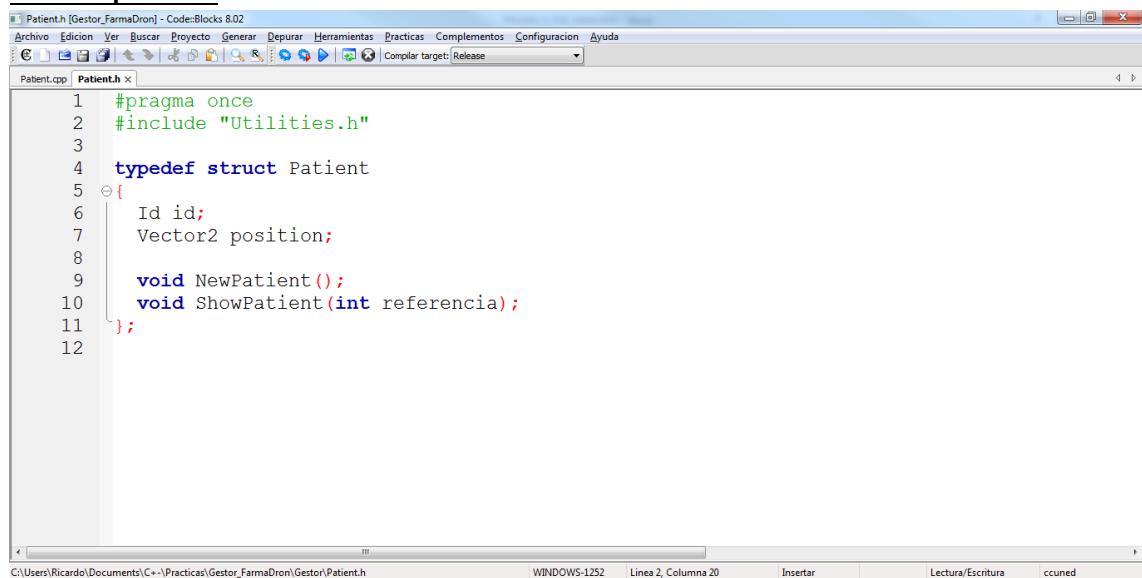
Objeto del módulo

Algunos datos y funciones, simples como para tener interés en el contexto de la aplicación, así como las importaciones de las librerías que permiten tratar con cadenas de caracteres y pedir y mostrar datos por pantalla se agrupan en este módulo.

Relación con otros módulos

Por lo básico de sus integrantes, es un módulo al que recurren el resto de los módulos de la aplicación.

Módulo paciente



The screenshot shows the Code::Blocks IDE interface with the Patient.h file open. The code defines a struct Patient with members Id and Vector2 position, and methods NewPatient() and ShowPatient(int referencia). The code is as follows:

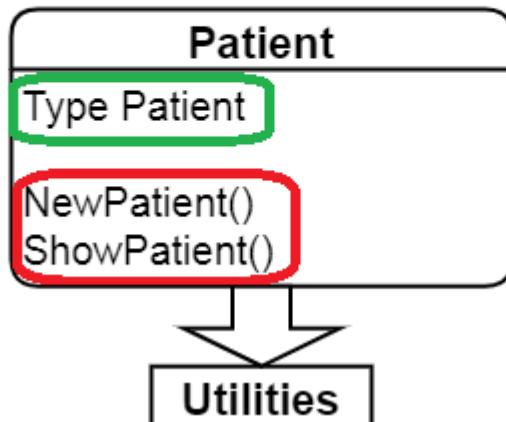
```
1 #pragma once
2 #include "Utilities.h"
3
4 typedef struct Patient
5 {
6     Id id;
7     Vector2 position;
8
9     void NewPatient();
10    void ShowPatient(int referencia);
11 };
12
```

File path: C:\Users\Ricardo\Documents\C+\Practicas\Gestor_FarmaDron\Gestor\Patient.h

Linea 2, Columna 20 Insertar Lectura/Escritura ccuned

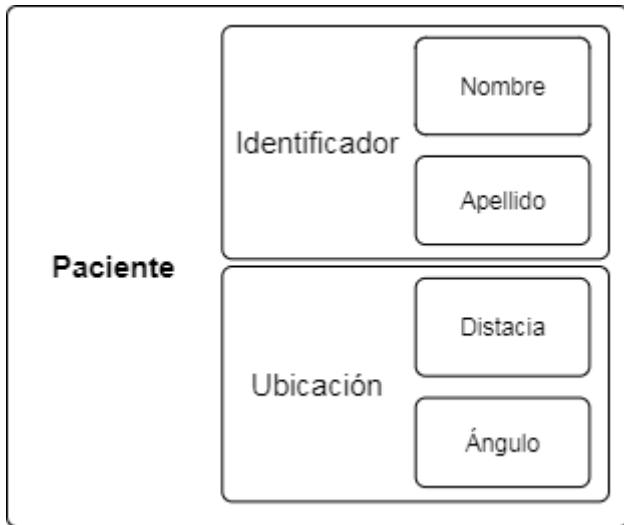
Elementos del módulo

Módulo Paciente



Tipo de dato

Tal y como los define la aplicación, un paciente tiene un *identificador* y una *ubicación*. El *identificador* está compuesto por el *nombre* y *apellido* del paciente y la *ubicación* por la *distancia* al almacén de referencia y el *ángulo* que forma la distancia con una dirección dada.



Cada uno de los campos del dato tipo *paciente* se puede identificar con tipos de datos creados en el módulo servicios y con tipos de datos básicos que reconoce y trata C++.

- **Nombre** → Cadena de caracteres. Tipo de dato definido en el módulo servicios.
- **Apellido** → Cadena de caracteres. Tipo de dato definido en el módulo servicios.
- **Distancia** → Número entero. Tipo de dato reconocido por C++.
- **Ángulo** → Número entero. Tipo de dato reconocido por C++.

Funciones

- **NewPatient.** La función inicializa la variable del dato tipo *paciente* preguntando por pantalla los datos de cada uno de los campos al usuario, discriminando datos que no tengan sentido para la aplicación como distancias fuera del radio de acción del Dron.

```

C:\Users\Ricardo\Documents\C++\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe

FarmaDron: Distribucion de Farmacos con Dron
  Alta nuevo paciente          <Pulsar A>
  Ubicar pacientes              <Pulsar U>
  Nuevo pedido                  <Pulsar N>
  Lista diario de pedidos      <Pulsar L>
  Calendario mensual de pedidos <Pulsar C>
  Salir                         <Pulsar S>
Teclear una opcion valida <A|U|N|L|C|S>? A
Alta nuevo paciente. Referencia paciente: 1
  Identificador <entre 1 y 20 caracteres>: Pedro Pérez
  Distancia <hasta 10000 metros a plena carga>: 50000
  Introdujo: 50000
  El FarmaDron no hace portes a mas de 15000 metros.
  Distancia <hasta 10000 metros a plena carga>: 5423
  Ángulo <entre 0 y 2000 pi / 1000 radianes>: -10
  Introdujo: -10
  Ángulo <entre 0 y 2000 pi / 1000 radianes>: 456
  Los datos son correctos? <S/N>?

  Distancia
  Ángulo
  
```

- **ShowPatient.** Muestra por pantalla los datos de la variable tipo *paciente* de acuerdo a un formato.

```
C:\Users\Ricardo\Documents\C+-\Prácticas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe
Los datos son correctos? <S/N> S
Otro paciente? <S/N> N
Desea volver al menu FarmáDron? <S/N> S
FarmáDron: Distribución de Farmacos con Dron
Alta nuevo paciente          <Pulsar A>
Ubicar pacientes              <Pulsar U>
Nuevo pedido                  <Pulsar N>
Lista diario de pedidos      <Pulsar L>
Calendario mensual de pedidos <Pulsar C>
Salir                         <Pulsar S>
Teclear una opción válida <A|U|N|L|C|S>? U
Lista de pacientes y su ubicación:
Ref.  Identificador          Distancia    Ángulo
1     Pedro Pérez            5423        456
2     María López             8876        1356
3     José Gómez              2789        867
4     Pablo García             1765        1823
5     Pilar González          11345       145
Desea volver al menu FarmáDron? <S/N>
```

Datos de la
variable tipo
paciente

Objeto del módulo

Crear el tipo de dato abstracto *paciente* básico para la aplicación y las funciones que manipulan la información contenida en variables declaradas con el tipo de dato *paciente*.

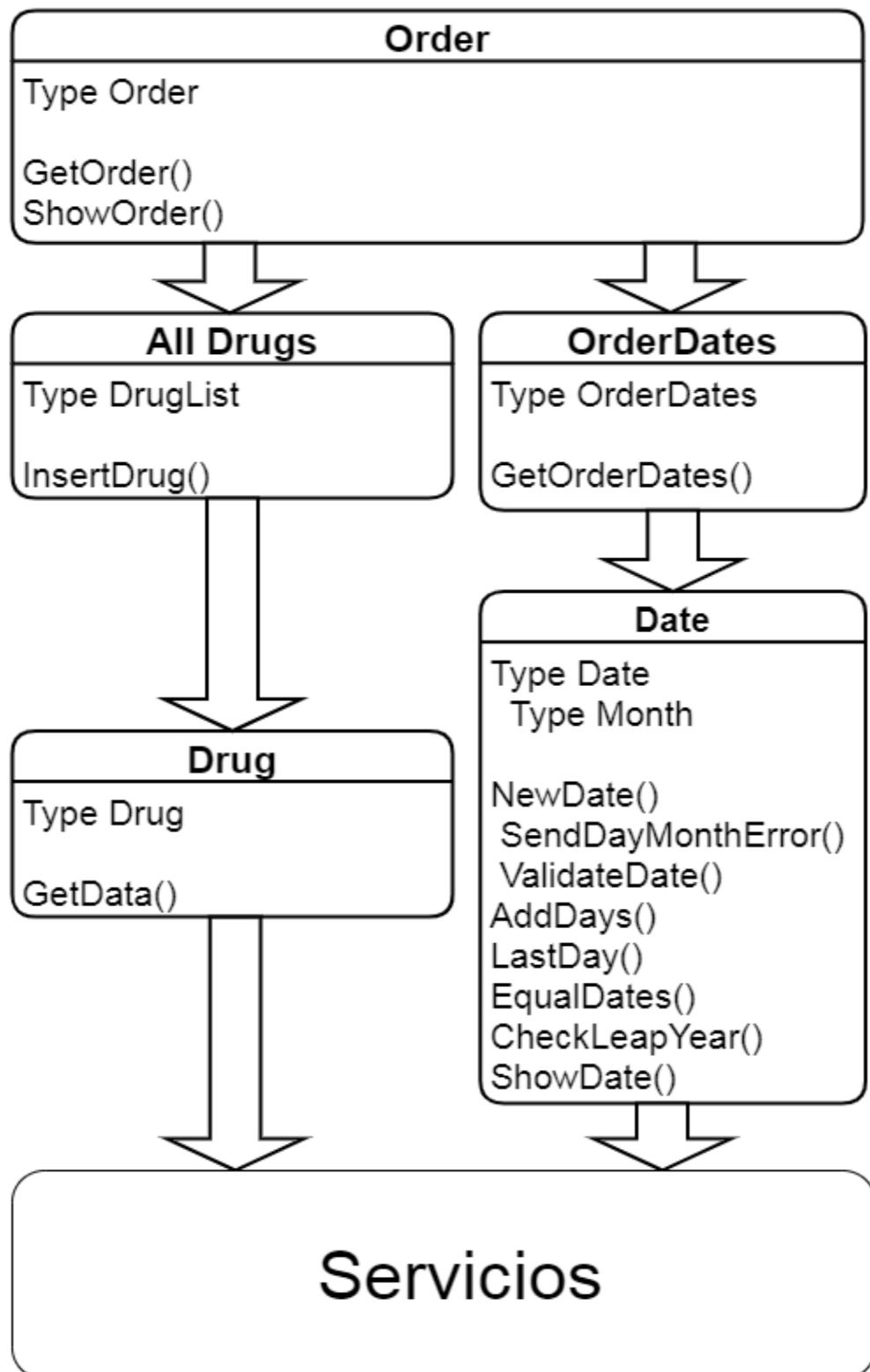
Relación con otros módulos

Salvo los módulos que forman parte de la construcción y manipulación del tipo de dato *pedido* complementario del tipo de dato *paciente* como tipo de datos básicos de la aplicación, así como los módulos dedicados a gestionar en exclusiva los datos de tipo *pedido*.

El módulo utiliza el módulo servicios.

Datos Tipo *Pedido*

Tipo de datos *Pedido*



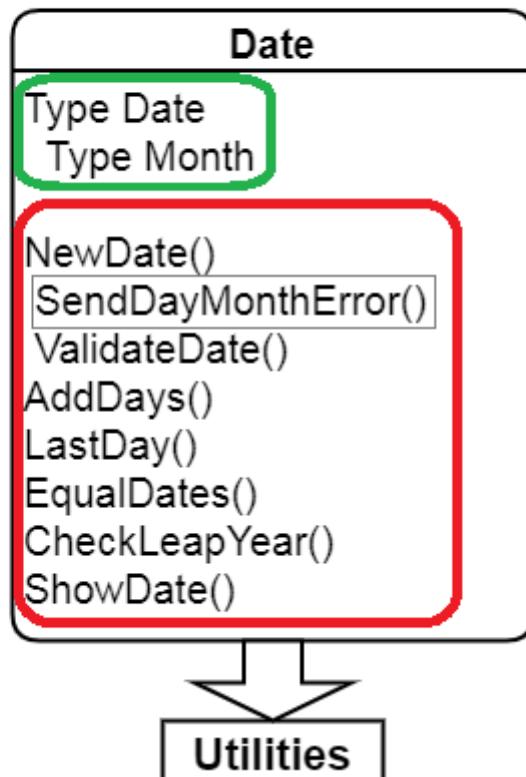
Módulo fecha

The screenshot shows the Code::Blocks IDE interface with the Date.h file open. The code defines an enum Month with values from january to december, and a struct Date with fields day, month, and year, along with various member functions like NewDate, Validate, AddDays, EqualDates, CheckLeapYear, and ShowDate.

```
1 #pragma once
2 #include "Utilities.h"
3
4 typedef enum Month
5 {
6     january,
7     february,
8     march,
9     april,
10    may,
11    june,
12    july,
13    august,
14    september,
15    october,
16    november,
17    december
18 };
19
20 typedef struct Date
21 {
22     int day;
23     Month month;
24     int year;
25
26     void NewDate(Text askDay, Text askMonth, Text askYear);
27     bool Validate();
28     int LastDay();
29     Date AddDays(int numberOfDays);
30     bool EqualDates(Date dateToCompare);
31     bool CheckLeapYear();
32     void ShowDate();
33 };
34
```

Elementos del módulo

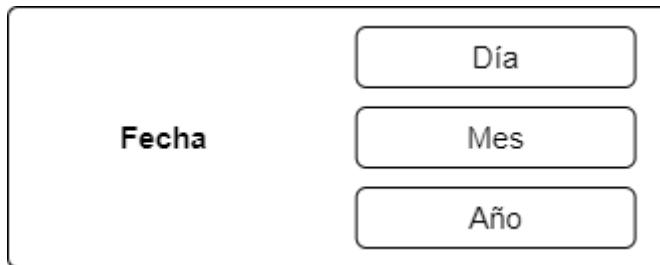
Módulo Fecha



La función recuadrada –SendDayMonthError– no es un subprograma que permite definir operaciones sobre datos tipo fecha, es una función que facilita la lectura e interpretación del código fuente en el fichero Date.cpp

Tipos de dato

Los campos que identifican el tipo de dato *fecha* son: día, mes y año.



Identificando cada uno de los campos que identifican el tipo de dato *fecha*:

- **Día** → Número entero. Tipo de dato reconocido por C++.
- **Mes** → Dato de tipo enumerado.

A screenshot of the Code::Blocks IDE interface. The title bar says "Date2.h [FarmaDron2] - Code::Blocks 8.02". The menu bar includes Archivo, Edición, Ver, Buscar, Proyecto, Generar, Depurar, Herramientas, Prácticas, Complementos, Configuración, and Ayuda. The toolbar has icons for file operations like New, Open, Save, and Build. The status bar at the bottom shows the path "C:\Users\Ricardo\Documents\C+\Prácticas\FarmaDron\FarmaDron2\Date2.h", the build configuration "WINDOWS-1252", "Línea 10, Columna 7", and buttons for Insertar, Lectura/Escritura, and ccuned. The code editor window displays the following C++ code:

```
4  typedef enum Month
5  {
6      january,
7      february,
8      march,
9      april,
10     may,
11     june,
12     july,
13     august,
14     september,
15     october,
16     november,
17     december
18 };
19
```

- **Año** → Número entero. Tipo de dato reconocido por C++.

Funciones

- **ValidateDate.** Chequea que los datos con los que se inicializa la variable tipo *fecha* tengan sentido en el formato de una fecha.
- **NewDate.** La función inicializa la variable del dato tipo *fecha* preguntando por pantalla los datos de cada uno de los campos al usuario, discriminando fechas que no tengan sentido.
 1. **SendDayMonthError.** Advierte al usuario mediante un mensaje por pantalla de que la fecha introducida no es correcta.

```
C:\Users\Ricardo\Documents\C++\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe
Otro pedido <S/N>? S
Pedido 2
    Ref. Paciente (entre 1 y 20) 1
    Numero de envios? 1
    Dia del envio? 40
    Mes del envio? 1
    Ano del envio? 2021
        Comprueba el número de días del mes
    Error. Introdujo dia: 40 y mes: 1.
    No se corresponde el numero de dias con el mes introducido
    Dia del envio? 20
    Mes del envio? 30
    Ano del envio? 2021
        Comprueba el número de meses del año
    Error. Introdujo el mes: 30
    Dia del envio?
```

- **LastDay** Calcula el último día del mes de la fecha que llama la función.
- **AddDays**. La función añade el número de días que se le pasan por parámetro a la fecha que contiene la variable devolviendo la fecha que resulta de añadir a la fecha inicial los días del parámetro.
- **EqualDates**. La función valora dos fechas decidiendo si son la misma.
- **CheckLeapYear**. La función valora si el año de la fecha es bisiesto.

Objeto del módulo

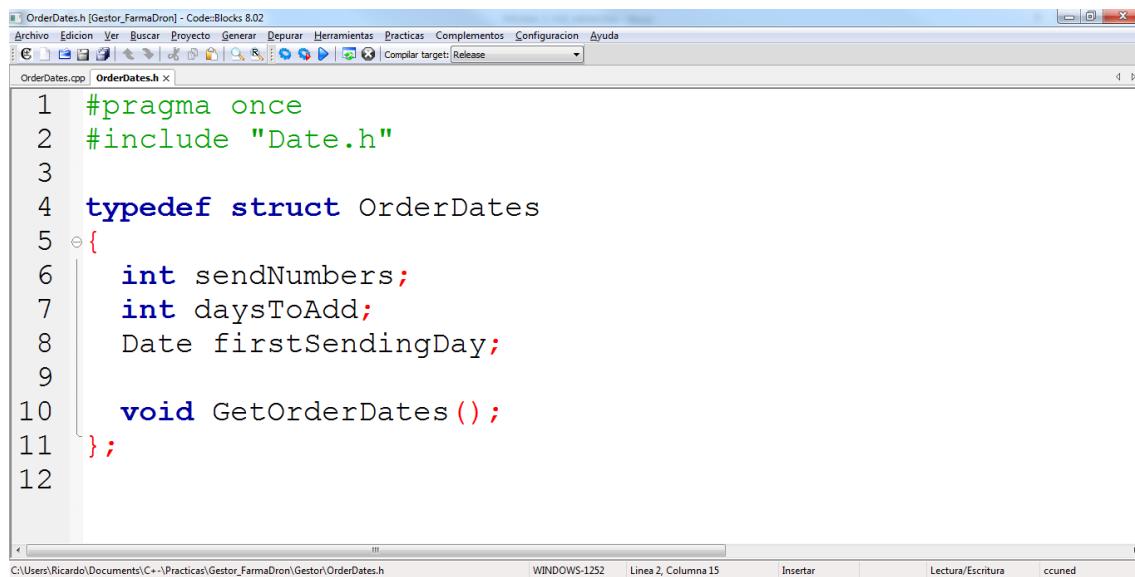
Poder trabajar con fechas resulta imprescindible en el contexto de una aplicación que gestiona pedidos; en muchos casos será el dato más importante y referencia del pedido, sin este módulo la aplicación no podría trabajar, exponer los pedidos que almacena ni diseñar rutas de reparto.

Relación con otros módulos

A este módulo recurrirán: el módulo que termina configurando el dato de tipo *pedido* y los módulos que gestionan los datos de los pedidos buscándolos por la fecha de envío.

El módulo utiliza el módulo servicios.

Módulo fechas del pedido

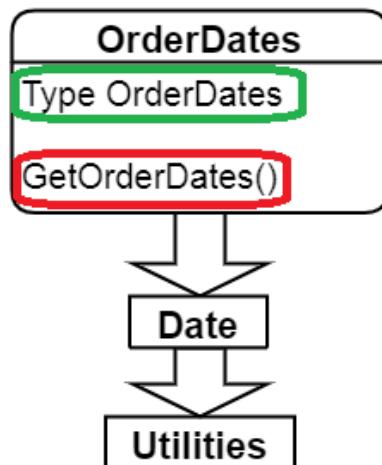


The screenshot shows the Code::Blocks IDE interface with the file OrderDates.h open. The code defines a struct named OrderDates with fields sendNumbers, daysToAdd, and firstSendingDay, and a method GetOrderDates(). The code is color-coded, with keywords in blue and identifiers in black.

```
1 #pragma once
2 #include "Date.h"
3
4 typedef struct OrderDates
5 {
6     int sendNumbers;
7     int daysToAdd;
8     Date firstSendingDay;
9
10    void GetOrderDates();
11 };
12
```

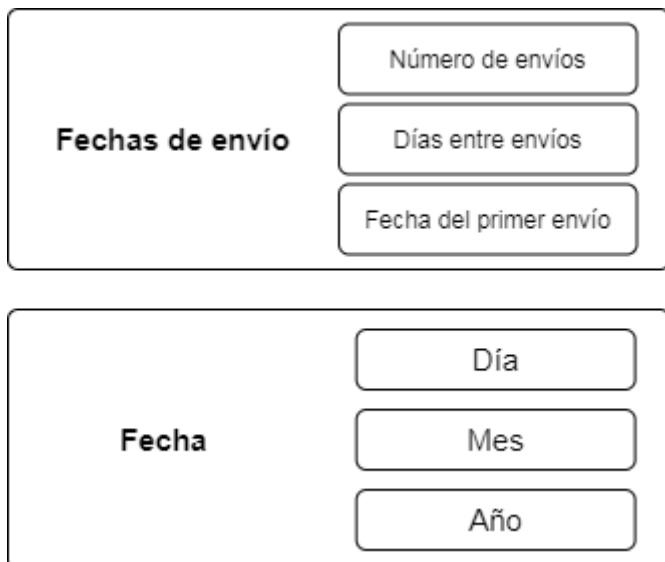
Elementos del módulo

Módulo Fechas del Pedido



Tipo de dato

El tipo de dato *fechas del pedido* se encuentra definido por los campos: número de envíos, días entre los envíos y fecha del primer envío según la estructura:



Identificando cada uno de los campos que identifican el tipo de dato *fecha*:

- **Número de envíos** → Número entero. Tipo de dato reconocido por C+-.
- **Días entre los envíos** → Número entero. Tipo de dato reconocido por C+-.
- **Fecha del primer envío** → Fecha. Tipo de dato definido en el módulo fecha.

Funciones

- **GetOrderDates** inicializa las variables del tipo de dato *fechas de envío* discriminando los valores que tengan sentido en el contexto de la aplicación para los campos del número de envíos y que los días entre los envíos no sean superiores a 15.

```

C:\Users\Ricardo\Documents\C++\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe

Angulo <entre 0 y 2000 pi / 1000 radianes>: 100
Los datos son correctos? <S/N>? S
Otro paciente? <S/N>? N
Desea volver al menu FarmaDron? <S/N> S
FarmaDron: Distribucion de Farmacos con Dron
  Alta nuevo paciente          <Pulsar A>
  Ubicar pacientes              <Pulsar U>
  Nuevo pedido                  <Pulsar N>
  Lista diario de pedidos      <Pulsar L>
  Calendario mensual de pedidos <Pulsar C>
  Salir                         <Pulsar S>
Teclear una opcion valida <A|U|N|I|L|C|S>? N
Pedido 1
Ref. Paciente <entre 1 y 20> 1
Numero de envios? -6
El sistema no puede procesar: -6 envios.
Numero de envios: -

```

```
C:\Users\Ricardo\Documents\C++\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe

Los datos son correctos? <S/N> S
Otro paciente? <S/N> N
Desea volver al menu FarmaDron? <S/N> S
FarmaDron: Distribucion de Farmacos con Dron
    Alta nuevo paciente          <Pulsar A>
    Ubicar pacientes              <Pulsar U>
    Nuevo pedido                  <Pulsar N>
    Lista diario de pedidos      <Pulsar L>
    Calendario mensual de pedidos <Pulsar C>
    Salir                         <Pulsar S>
Teclear una opcion valida <A|U|N|I|L|C|S>? N
Pedido 1
    Ref. Paciente <entre 1 y 20> 1
    Numero de envios? 2
    Numero de dias entre cada envio <Entre 1 y 15 dias>? 17
    El sistema no procesa 17 dias entre envios
    Numero de dias entre cada envio <Entre 1 y 15 dias>? -
```

Objeto del módulo

Los pedidos pueden tener envíos periódicos; éste módulo permite, a partir del dato *fecha* del módulo anterior, definir el tipo de dato que gestiona la periodicidad de los pedidos.

Relación con otros módulos

El tipo de dato que proporciona este módulo se puede considerar una mejora y adaptación a las necesidades reales de la aplicación, así a éste módulo recurrirán todos los módulos que necesiten trabajar con las fechas.

El módulo utiliza los módulos servicios y fecha.

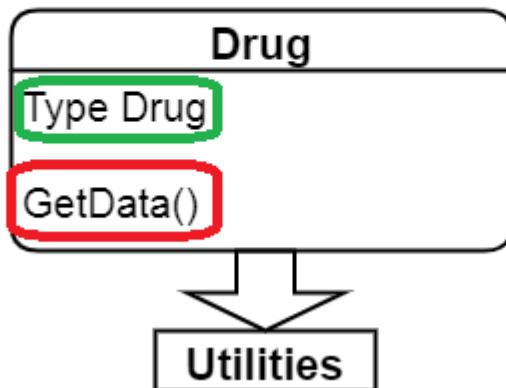
Módulo fármaco

The screenshot shows the Code::Blocks 8.02 IDE interface. The title bar says "Drug.h [Gestor_FarmaDron] - Code::Blocks 8.02". The menu bar includes Archivo, Edición, Ver, Buscar, Proyecto, Generar, Depurar, Herramientas, Prácticas, Complementos, Configuración, Ayuda. The toolbar has icons for file operations like Open, Save, Print, and Build. The status bar at the bottom shows the path "C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Drug.h", the build configuration "WINDOWS-1252", line "Línea 2, Columna 20", and other settings.

```
1 #pragma once
2 #include "Utilities.h"
3
4 typedef struct Drug {
5
6     Text name;
7     int weight;
8     int unities;
9
10    void GetData();
11 };
12
```

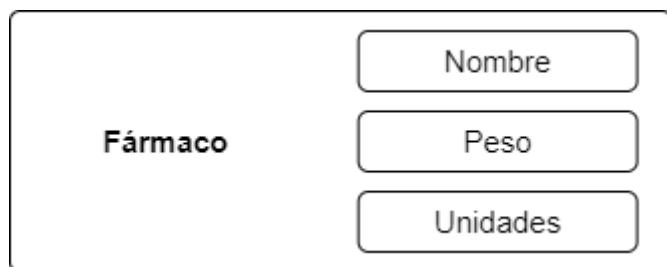
Elementos del módulo

Módulo Fármaco



Tipo de dato

El tipo de dato *fármaco* se encuentra definido por los campos: nombre, peso y unidades.



Identificando cada uno de los campos que identifican el tipo de dato *fármaco*:

- **Nombre** → Texto. Tipo de dato definido en el módulo servicios.
- **Peso** → Número entero. Tipo de dato reconocido por C++.

- **Unidades** → Número entero. Tipo de dato reconocido por C++.

Funciones

- **GetData.** La función inicializa la variable del dato tipo *fármaco* preguntando por pantalla al usuario los datos que identifican el dato de tipo *fármaco*, discriminando datos que no tengan sentido para la aplicación como pesos del fármaco superiores a la capacidad de transporte del dron.

```

C:\Users\Ricardo\Documents\C++\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe

Ref. Paciente <entre 1 y 20> 3
Número de envíos? 1
Dia del envío? 23
Mes del envío? 1
Año del envío? 2021
Nombre del farmaco <Entre 1 y 20 caracteres>? Analgésico
Peso farmaco <Menor de 3000 gramos>? 3050
No se transporta peso igual o superior a 3000 gramos.

Peso farmaco <Menor de 3000 gramos>? 1000
Unidades de farmaco? 4
Introdujo: 4 unidades * 1000 peso por unidad. Resulta: 4000 gramos.
El sistema no transporta mas de 3000 gramos.

Peso farmaco <Menor de 3000 gramos>? 

```

Objeto del módulo

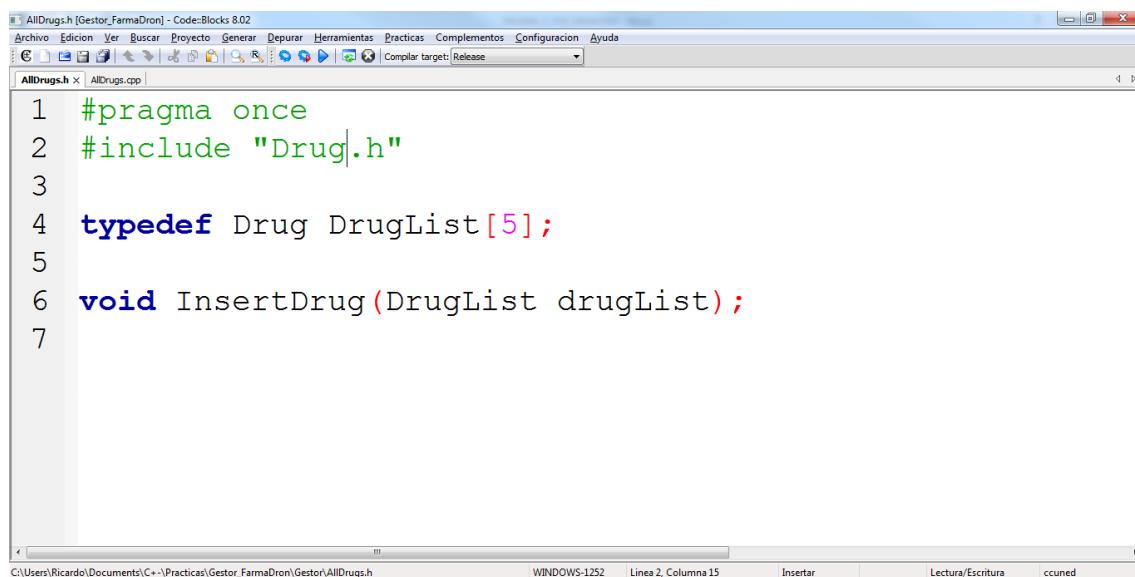
La aplicación gestiona el envío de medicamentos a pacientes; el tipo de dato *fármaco* resulta básico en la definición de un pedido; es el objeto del pedido.

Relación con otros módulos

Este módulo será utilizado por cualquier módulo que tenga relación con los pedidos. Salvo los módulos que trabajan con los datos de tipo *paciente* y los módulos de los datos tipo *pedido* encargados de la gestión de las fechas y la periodicidad de los envíos, todos los módulos de la aplicación tendrán relación con él.

El módulo utiliza el [módulo servicios](#).

Módulo lista de fármacos

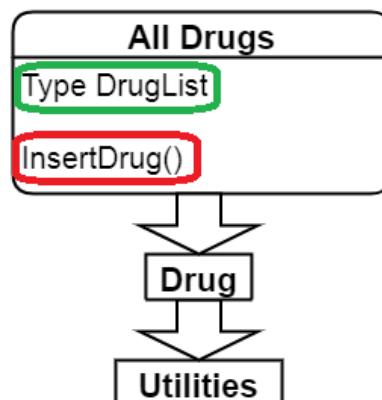


```
1 #pragma once
2 #include "Drug.h"
3
4 typedef Drug DrugList[5];
5
6 void InsertDrug (DrugList drugList);
```

The screenshot shows the Code::Blocks IDE interface with the file 'AllDrugs.h' open. The code editor contains the provided C++ code. The file path 'C:\Users\Ricardo\Documents\C++\Practicas\Gestor_FarmaDron\Gestor\AllDrugs.h' is visible at the bottom left. The status bar at the bottom right shows 'WINDOWS-1252', 'Linea 2, Columna 15', 'Insertar', 'Lectura/Escritura', and 'ccuned'. The menu bar includes Archivo, Edición, Ver, Buscar, Proyecto, Generar, Depurar, Herramientas, Prácticas, Complementos, Configuración, Ayuda.

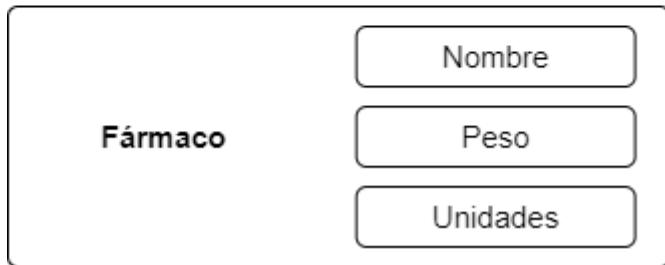
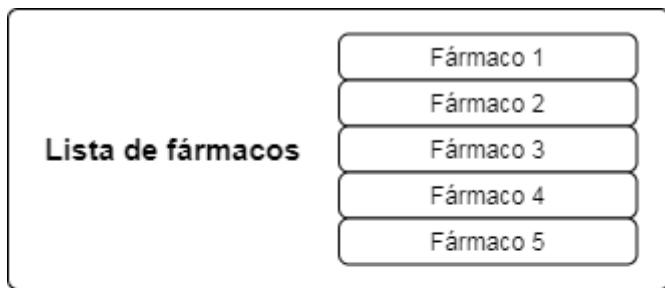
Elementos del módulo

Módulo Lista de Fármacos



Tipo de dato

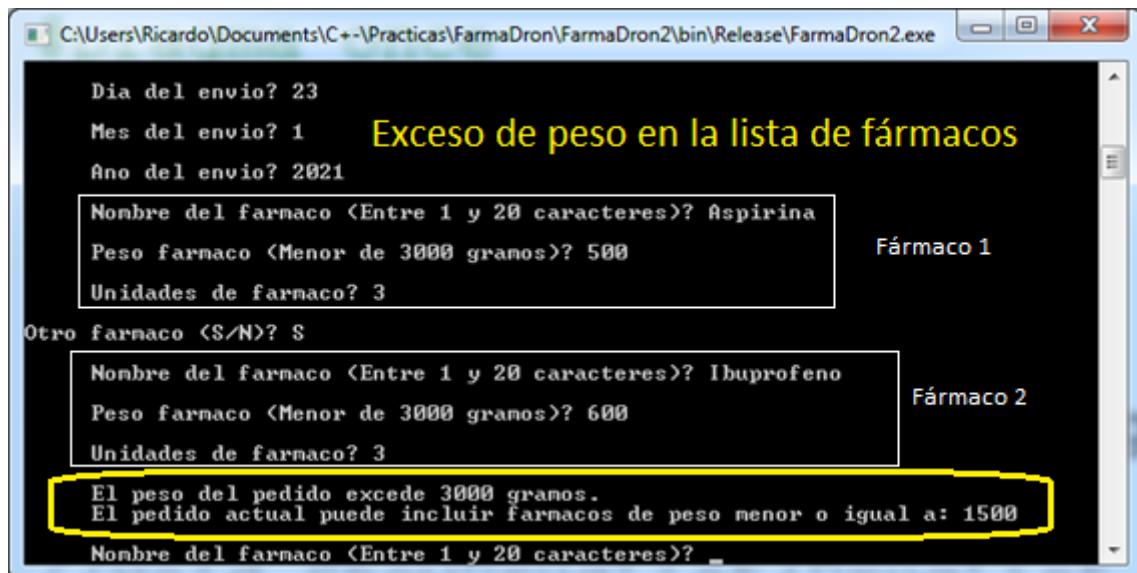
El tipo de dato *lista de fármacos* es un array de 5 datos de tipo *fármaco*.

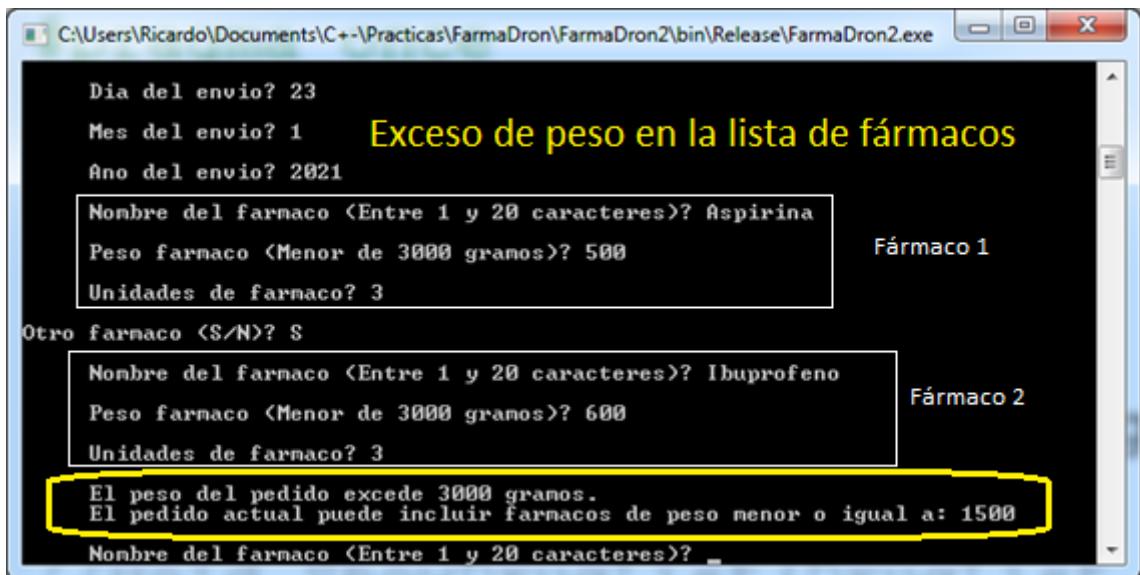


El tipo de dato **fármaco** se encuentra definido en el módulo anterior.

Funciones

- **InsertDrug.** Registra los fármacos del pedido evitando que el peso total de los fármacos supere el peso que puede transportar el dron y advirtiendo al usuario cuando el número de fármacos es superior a cinco, máxima capacidad fijada en la aplicación.





Objeto del módulo

Permitir que los pedidos tengan el formato que requiere la aplicación, es decir que se puedan hacer pedidos de 5 fármacos para un mismo envío.

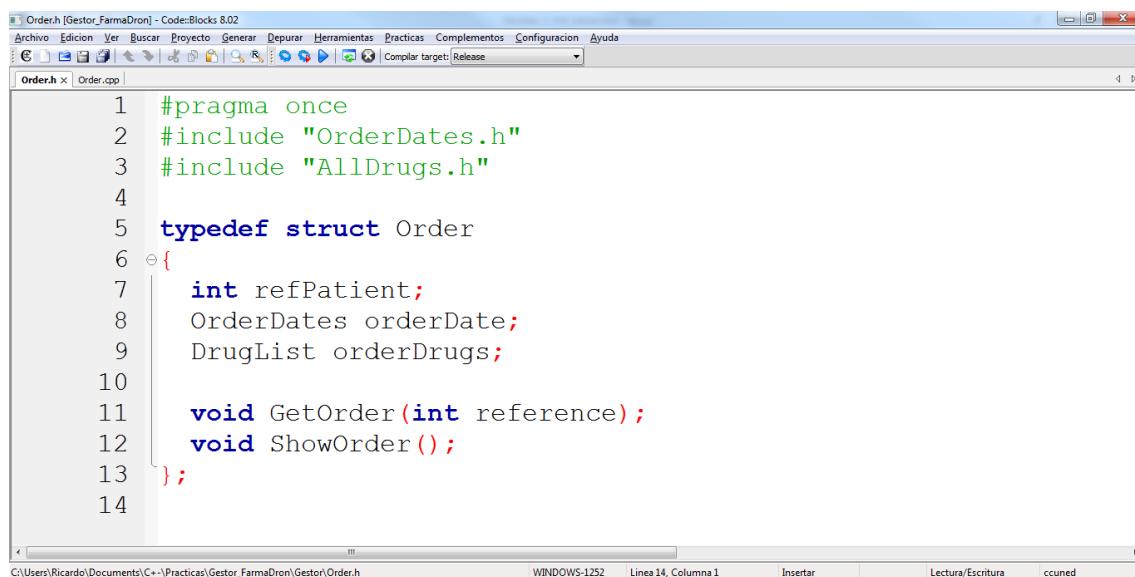
Relación con otros módulos

Cualquier pedido hace referencia a este módulo por más que sólo se envíe un fármaco. En especial se recurre a la información que guardan datos de este tipo cuando se trabaja con el peso del envío.

Los módulos que trabajan con los datos de tipo *paciente* o *fecha* de los pedidos son los únicos módulos que no necesitan relacionarse con él.

El módulo utiliza los módulos servicios y fármaco.

Módulo pedidos

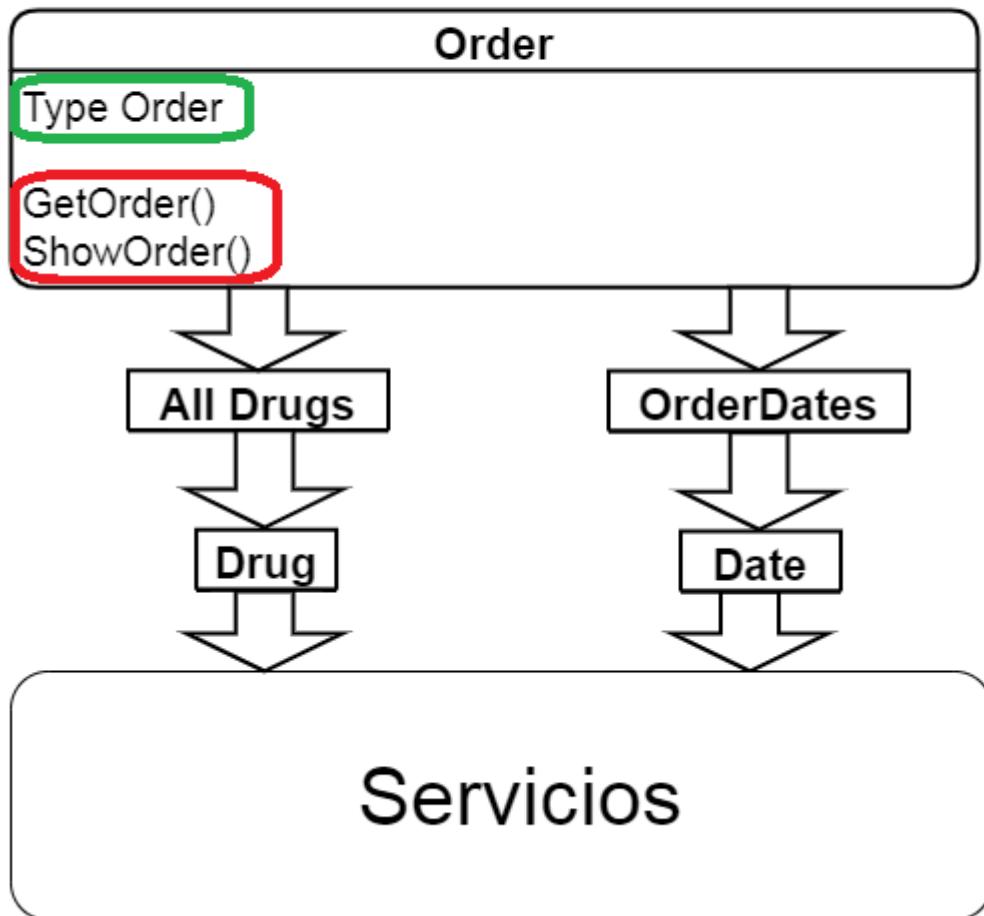


```
Order.h [Gestor_FarmaDron] - Code::Blocks 8.02
Archivo Edición Ver Buscar Proyecto Generar Depurar Herramientas Prácticas Complementos Configuración Ayuda
C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Order.h Comilar target: Release
Order.h x Order.cpp
1 #pragma once
2 #include "OrderDates.h"
3 #include "AllDrugs.h"
4
5 typedef struct Order
6 {
7     int refPatient;
8     OrderDates orderDate;
9     DrugList orderDrugs;
10
11     void GetOrder(int reference);
12     void ShowOrder();
13 };
14
```

C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Order.h WINDOWS-1252 Línea 14, Columna 1 Insertar Lectura/Escritura ccuned

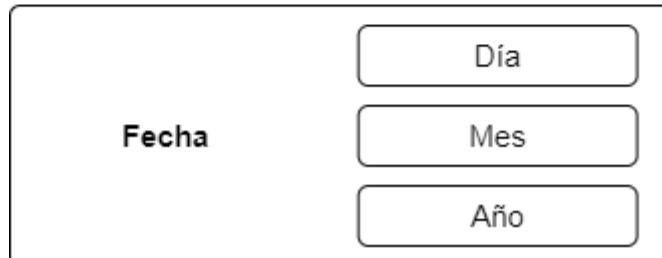
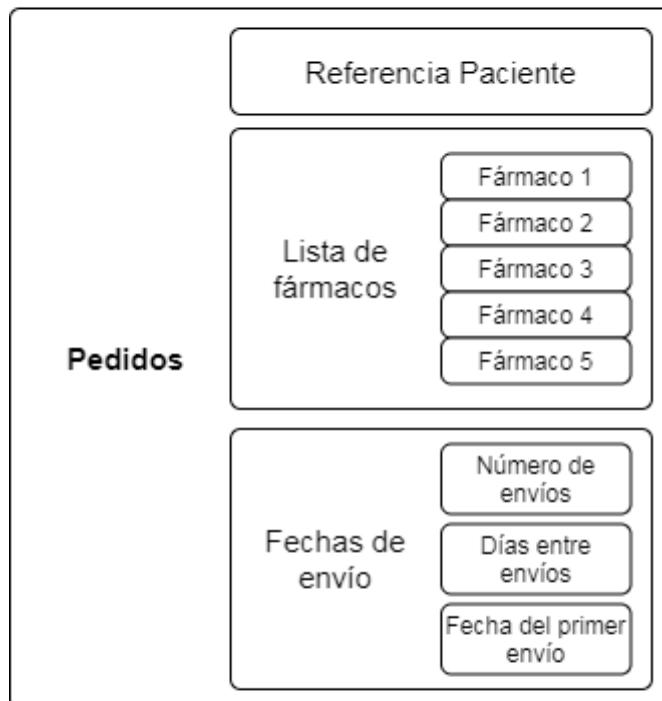
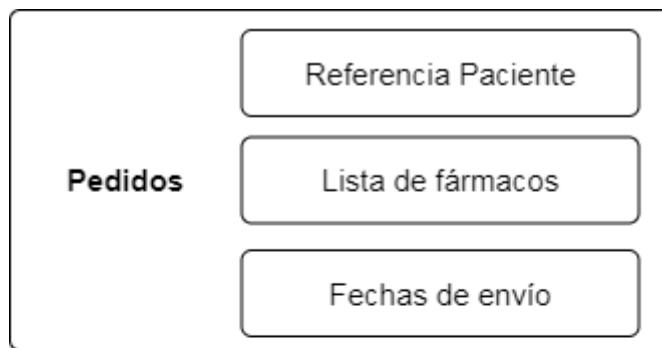
Elementos del módulo

Módulo Pedidos



Tipo de dato

El tipo de dato *pedido* está formado por los datos: referencia paciente, lista de fármacos y fechas de envío.

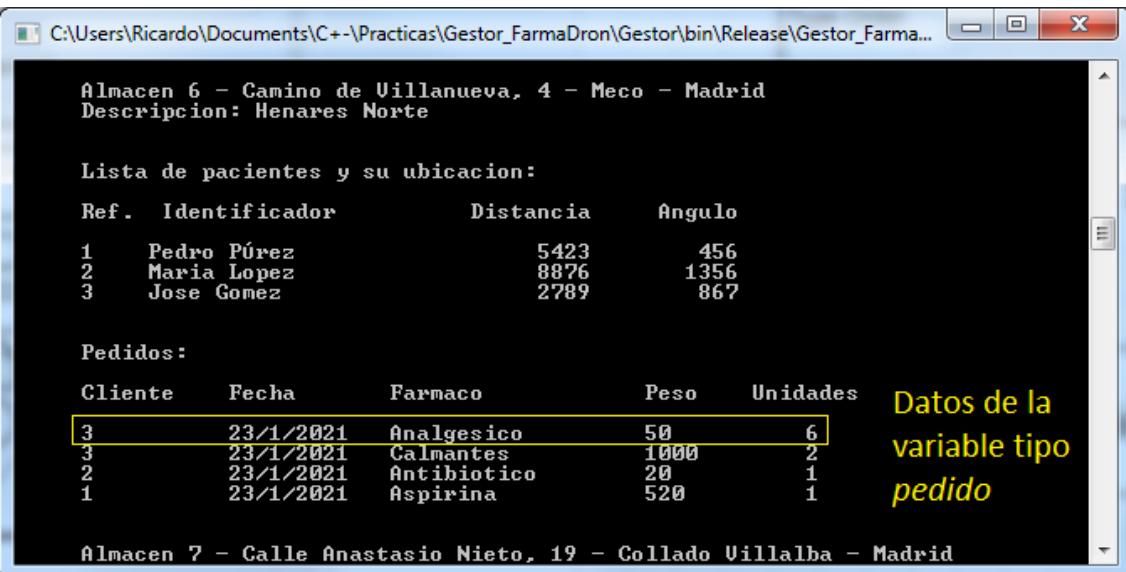


- **Referencia Paciente** → Número entero. Tipo de dato reconocido por C+-.

- **Lista de fármacos** y **Fechas de envío** están descritos en los módulos anteriores.

Funciones

- **GetOrder**. La función inicializa la variable del dato tipo *pedido* preguntando por pantalla los datos necesarios hasta haber completado la información del pedido.
- **ShowOrder**. Muestra por pantalla los datos de un pedido de acuerdo a un determinado formato.



The screenshot shows a Windows application window titled "C:\Users\Ricardo\Documents\C++\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...". Inside the window, there is a text-based interface:

```

Almacen 6 - Camino de Villanueva, 4 - Meco - Madrid
Descripción: Henares Norte

Lista de pacientes y su ubicación:
Ref. Identificador          Distancia      Ángulo
1   Pedro Pérez              5423           456
2   María López              8876           1356
3   José Gómez               2789           867

Pedidos:
Cliente  Fecha        Farmaco     Peso  Unidades
3         23/1/2021  Analgesico   50    6
3         23/1/2021  Calmantes    1000   2
2         23/1/2021  Antibiotico  20    1
1         23/1/2021  Aspirina     520   1

```

A yellow box highlights the first row of the "Pedidos" table, specifically the columns "Cliente", "Fecha", "Farmaco", "Peso", and "Unidades". To the right of this highlighted area, the text "Datos de la variable tipo pedido" is written in yellow.

At the bottom of the window, it says "Almacen 7 - Calle Anastasio Nieto, 19 - Collado Villalba - Madrid".

Objeto del módulo

Con este módulo se completa la definición de los datos de tipo *pedido* y gracias a él se podrá trabajar con los pedidos tal y como son definidos por la aplicación.

Relación con otros módulos

Cualquier módulo que necesite trabajar con un dato de un pedido recurrirá a este módulo.

El módulo utiliza los módulos servicios, fármaco, lista de fármacos, fecha y fechas de envío.

Gestión del Paquete *Almacén*

Gestión de Paquete *Almacén*

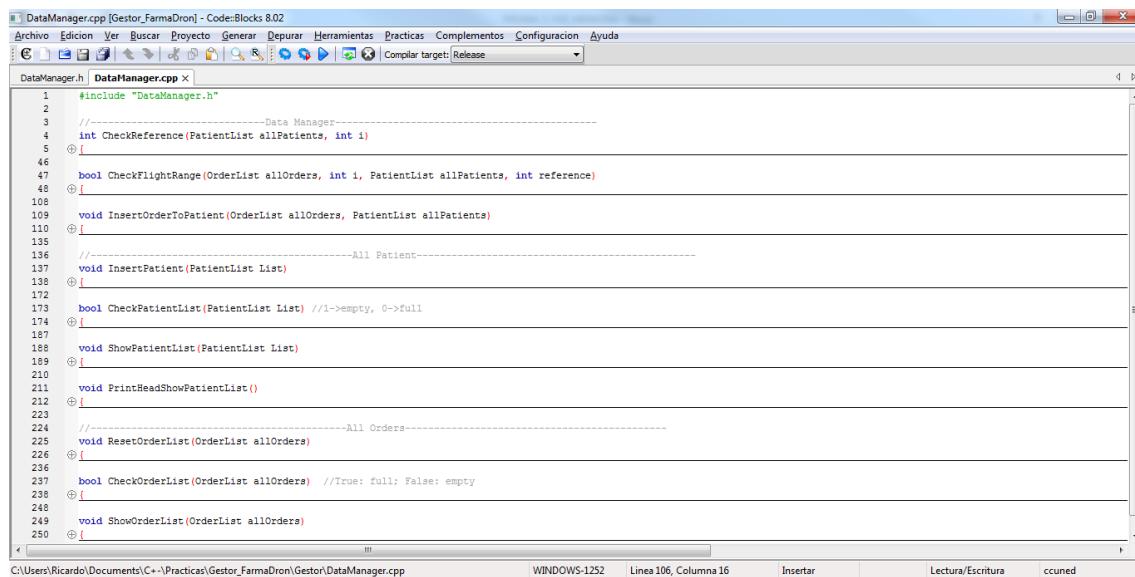
Gestión de Datos

Datos Pedidos

Datos Pacientes

Servicios

Módulo gestor de datos

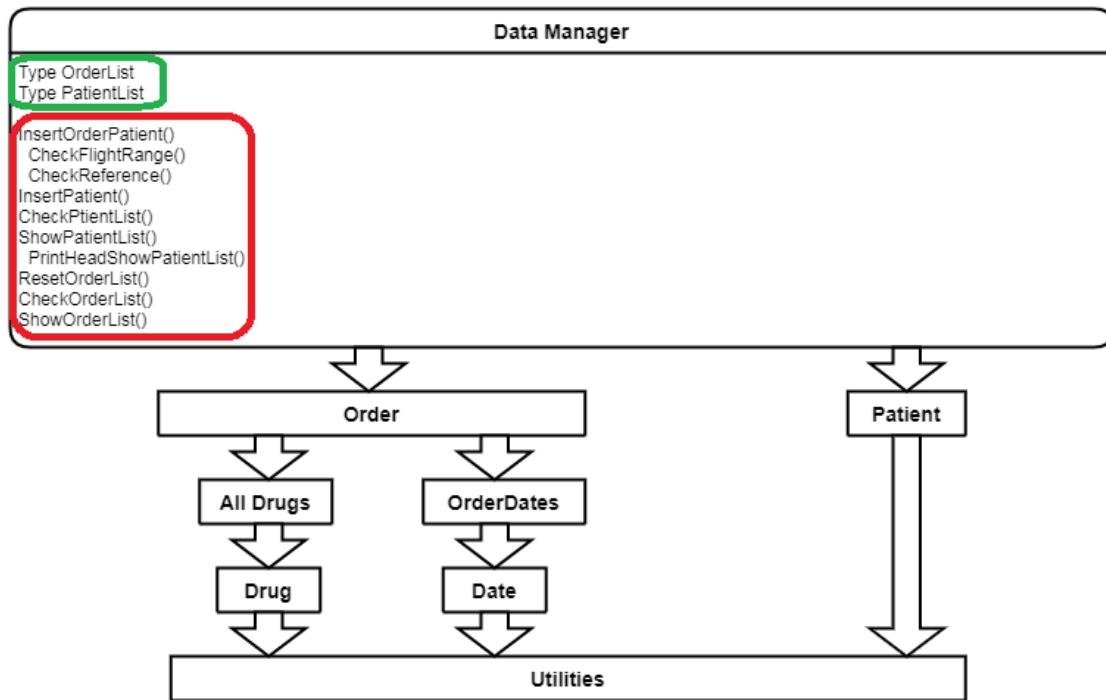


The screenshot shows the Code::Blocks IDE interface with the file 'DataManager.cpp' open. The code is written in C++ and defines a class 'DataManager' with various methods for managing patient and order lists. The code includes comments and several functions like InsertOrderToPatient, CheckFlightRange, and ShowPatientList.

```
1 #include "DataManager.h"
2
3 //-----Data Manager-----
4 int CheckReference(PatientList allPatients, int i)
5 {
6
7     bool CheckFlightRange(OrderList allOrders, int i, PatientList allPatients, int reference)
8     {
9
10         void InsertOrderToPatient(OrderList allOrders, PatientList allPatients)
11     {
12
13         //-----All Patient-----
14         void InsertPatient(PatientList List)
15     {
16
17         bool CheckPatientList(PatientList List) //l->empty, 0->full
18
19         void ShowPatientList(PatientList List)
20     {
21
22         void PrintHeadShowPatientList()
23     {
24
25         //-----All Orders-----
26         void ResetOrderList(OrderList allOrders)
27     {
28
29         bool CheckOrderList(OrderList allOrders) //True: full; False: empty
30     {
31
32         void ShowOrderList(OrderList allOrders)
33     {
34 }
```

Elementos del módulo

Módulo Gestión de Datos



Tipos de dato

En este módulo se definen dos tipos de datos:

- **Array de datos tipo paciente.** Tipo de dato definido en los módulos anteriores.
- **Array de datos tipo pedido.** Tipo de dato definido en los módulos anteriores.

Funciones

- **InsertOrderToPatient.** A la hora de registra los pedidos, algunos campos del dato tipo *Paciente* y del dato tipo *Pedido* tienen relación. Esta función comprueba que las relaciones entre los distintos campos son correctas.
 - **CheckReference.** Comprueba la referencia de paciente al que se envía el pedido contiene datos y no se está enviando el pedido a pacientes inexistentes.

```

C:\Users\Ricardo\Documents\C-\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe

-- 20 21 22 | 23 24
25 26 27 28 29 | -- --
Desea volver al menu FarmaDron? <S/N> S
FarmaDron: Distribucion de Farmacos con Dron
  Alta nuevo paciente          <Pulsar A>
  Ubicar pacientes             <Pulsar U>
  Nuevo pedido                 <Pulsar N>
  Lista diario de pedidos     <Pulsar L>
  Calendario mensual de pedidos <Pulsar C>
  Salir                         <Pulsar S>
Teclear una opcion valida <A|U|N|L|C|S>? N
Pedido 3
  Ref. Paciente <entre 1 y 20> 6
  Introdujo como referencia paciente: 6
  No existe ningun paciente registrado con esa referencia.

Pedido 3
  Ref. Paciente <entre 1 y 20> 5
  Numero de envios? _
```

- **CheckFlightRange.** Las distancias a las que se encuentran algunos pacientes pueden estar dentro del radio de acción del dron, pero no del radio de acción del dron para cualquier peso que tenga el pedido. Esta función comprueba siguiendo una sencilla relación lineal ($autonomía = (25000 - 5/3 \times peso\ total\ del\ pedido) / 2$) entre peso del pedido y autonomía del dron que los datos sean correctos.

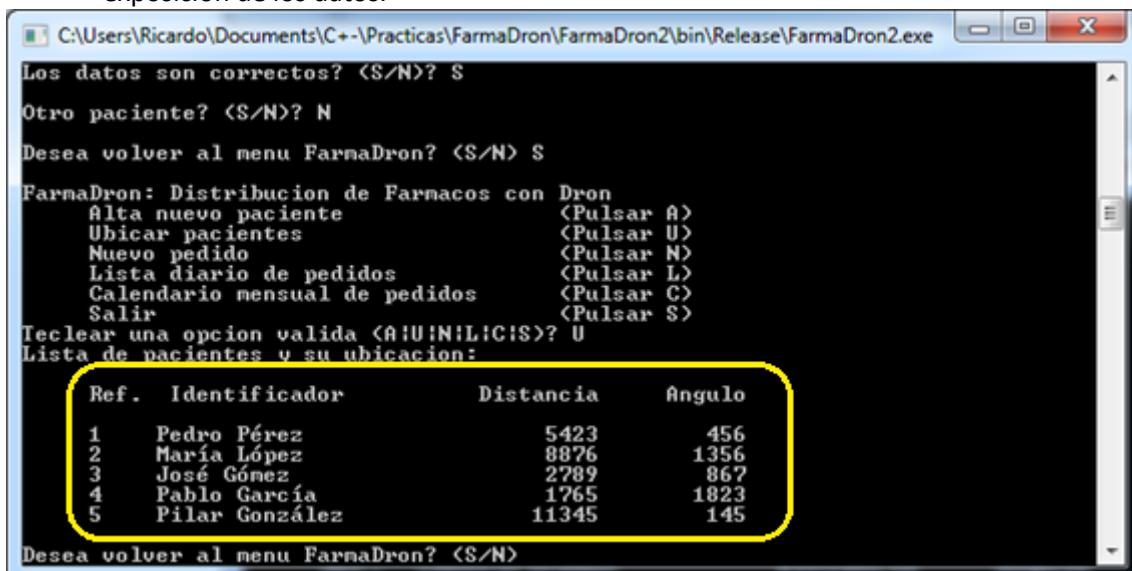
```

C:\Users\Ricardo\Documents\C-\Practicas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe

  Unidades de farmaco? 2
  Otro farmaco <S/N>? S
    Nombre del farmaco <Entre 1 y 20 caracteres>? Paracetamol
    Peso farmaco <Menor de 3000 gramos>? 250
    Unidades de farmaco? 3
  Otro farmaco <S/N>? N
    *** LA APLICACION NO HA REGISTRADO EL PEDIDO ***
    =====
    El peso del pedido es de 2750 gramos.
    A ese peso le corresponde una autonomia de vuelo de: 10208.3 metros.
    El paciente se encuentra a una distancia: 12000.0 metros
    El paciente esta fuera de la autonomia de vuelo del dron
    ** SE REPITE LA TOMA DE DATOS DESDE EL PRINCIPIO **
    Por favor, incluya menos peso en el pedido
Pedido 1
  Ref. Paciente <entre 1 y 20>
```

- **InsertPatient.** La función recorre la lista ordenada de pacientes inicializando las variables que no contienen datos. Para inicializar una variable es suficiente con las funciones que se declararon en el dato de tipo *paciente*.
- **CheckPatientList.** La función comprueba si un array de datos paciente ha sido inicializado.

- **ShowPatientList.** Esta función expone todos los pacientes que están registrados en la aplicación.
 - **PrintHeadShowPatientList.** Imprime un encabezamiento que da formato a la exposición de los datos.



The screenshot shows a terminal window titled 'C:\Users\Ricardo\Documents\C++\Prácticas\FarmaDron\FarmaDron2\bin\Release\FarmaDron2.exe'. The window displays the following text:

```

Los datos son correctos? <S/N> S
Otro paciente? <S/N> N
Desea volver al menu FarmaDron? <S/N> S
FarmaDron: Distribucion de Farmacos con Dron
  Alta nuevo paciente          <Pulsar A>
  Ubicar pacientes             <Pulsar U>
  Nuevo pedido                 <Pulsar N>
  Lista diario de pedidos     <Pulsar L>
  Calendario mensual de pedidos <Pulsar C>
  Salir                        <Pulsar S>
Teclear una opcion valida <A|U|N|L|C|S>? U
Lista de pacientes y su ubicacion:
  Ref. Identificador           Distancia    Ángulo
  1   Pedro Pérez               5423        456
  2   María López               8876        1356
  3   José Gómez                2789        867
  4   Pablo García              1765        1823
  5   Pilar González            11345       145
Desea volver al menu FarmaDron? <S/N>

```

A yellow box highlights the table of patients and their locations.

Ref.	Identificador	Distancia	Ángulo
1	Pedro Pérez	5423	456
2	María López	8876	1356
3	José Gómez	2789	867
4	Pablo García	1765	1823
5	Pilar González	11345	145

- **ResetOrderList.** Inicializa todas las variables a valores que no tienen sentido en el contexto de la aplicación. Equivale a un reseteo de la lista de pedidos, cuando la aplicación encuentra en uno de los pedidos valores iguales a los del reseteo, tratará la variable como si estuviera vacía.
- **CheckOrderList.** La función comprueba si un array de datos pedido ha sido inicializado.
- **ShowOrderList.** Muestra los pedidos registrados en un array de datos pedido.

Objeto del módulo

En el contexto de la aplicación debe existir concordancia entre los valores que se registran en un tipo de dato pedido y un dato de tipo paciente que estén relacionados. La funciones de este módulo, así como el tipo de datos que definen, hace que los datos que se definieron en los módulos anteriores tengan sentido en el contexto general de la aplicación para permitir trabajar con ellos de forma realista y no como colecciones de número y caracteres inconexas.

Relación con otros módulos

Cualquier módulo que registre o exponga los dos datos que se han definido como fundamentales para la aplicación –datos pedido y datos paciente- recurrirá a las funciones de este módulo.

El módulo utiliza los módulos servicios, fármaco, lista de fármacos, fecha, fechas de envío y pacientes.

Gestión de Almacenes

Gestión de Almacenes

Gestores de Almacenes



Gestor Almacenes

Gestión de
Datos

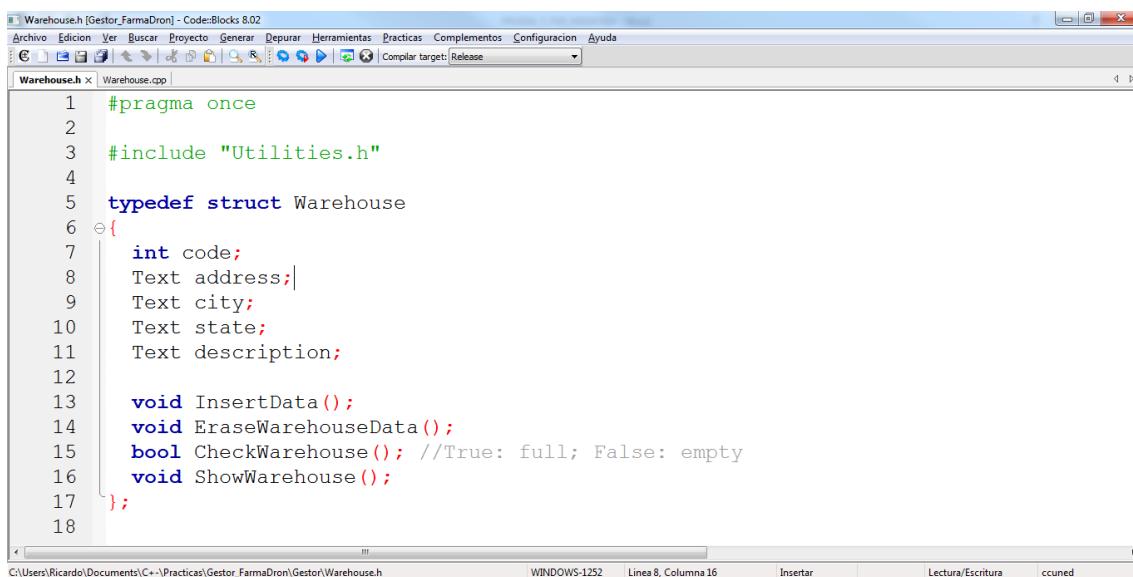
Dato Almacén

Paquete
Almacén



Servicios

Módulo almacén

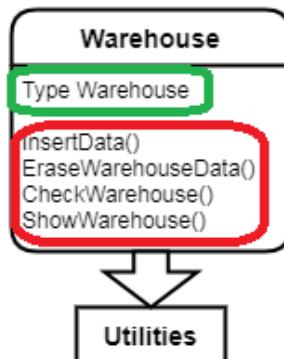


```
Warehouse.h [Gestor_FarmaDron] - Code::Blocks 8.02
Archivo Edición Ver Buscar Proyecto Generar Depurar Herramientas Prácticas Complementos Configuración Ayuda
C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Warehouse.h Comilar target: Release
Warehouse.h x Warehouse.cpp
1 #pragma once
2
3 #include "Utilities.h"
4
5 typedef struct Warehouse
6 {
7     int code;
8     Text address;
9     Text city;
10    Text state;
11    Text description;
12
13    void InsertData();
14    void EraseWarehouseData();
15    bool CheckWarehouse(); //True: full; False: empty
16    void ShowWarehouse();
17 };
18
```

C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\Warehouse.h WINDOWS-1252 Línea 8, Columna 16 Insertar Lectura/Escritura ccuned

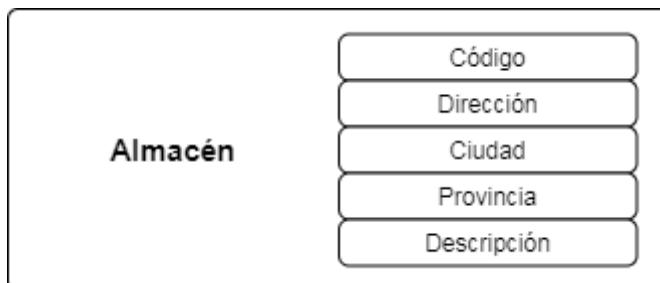
Elementos del módulo

Módulo Almacén



Tipo de dato

El tipo de dato *almacén* está formado por los datos: código, dirección, ciudad, provincia y descripción.



- **Código** → Número entero. Tipo de dato reconocido por C++.
- **Dirección** → Texto. Tipo de dato definido en el módulo servicios.
- **Ciudad** → Texto. Tipo de dato definido en el módulo servicios.

- **Provincia** → Texto. Tipo de dato definido en el módulo servicios.
- **Descripción** → Texto. Tipo de dato definido en el módulo servicios.

Funciones

- **InsertData**. Inicializa la variable de tipo *almacén* preguntando por pantalla los datos de cada uno de los campos al usuario.
- **EraseWarehouseData**. Da un código inexistente al almacén. La aplicación tratará como almacenes vacíos a los almacenes con el código que proporciona esta función.
- **CheckWarehouseData**. Examina el código del almacén para saber si se trata de un almacén vacío y devuelvo un valor verdadero o falso según el estado.
- **ShowWarehouseData**. Muestra por pantalla la información de la variable de tipo *almacén* con un formato.

```

Almacen 7 - Calle Anastasio Nieto, 19 - Collado Villalba - Madrid
Descripcion: Guadarradama

Lista de pacientes y su ubicacion:

Ref. Identificador Distancia Angulo
1 Cesar Bastos 1300 125
2 Ana Loureiro 906 1875
3 Ruben Parrilla 450 1050
4 Joaquin Ruiz 1000 1475

Pedidos:

Cliente Fecha Farmaco Peso Unidades
1 28/10/2021 Colutorio 300 1
3 28/10/2021 Jarabe 100 3
4 28/10/2021 Panales 500 1

```

Objeto del módulo

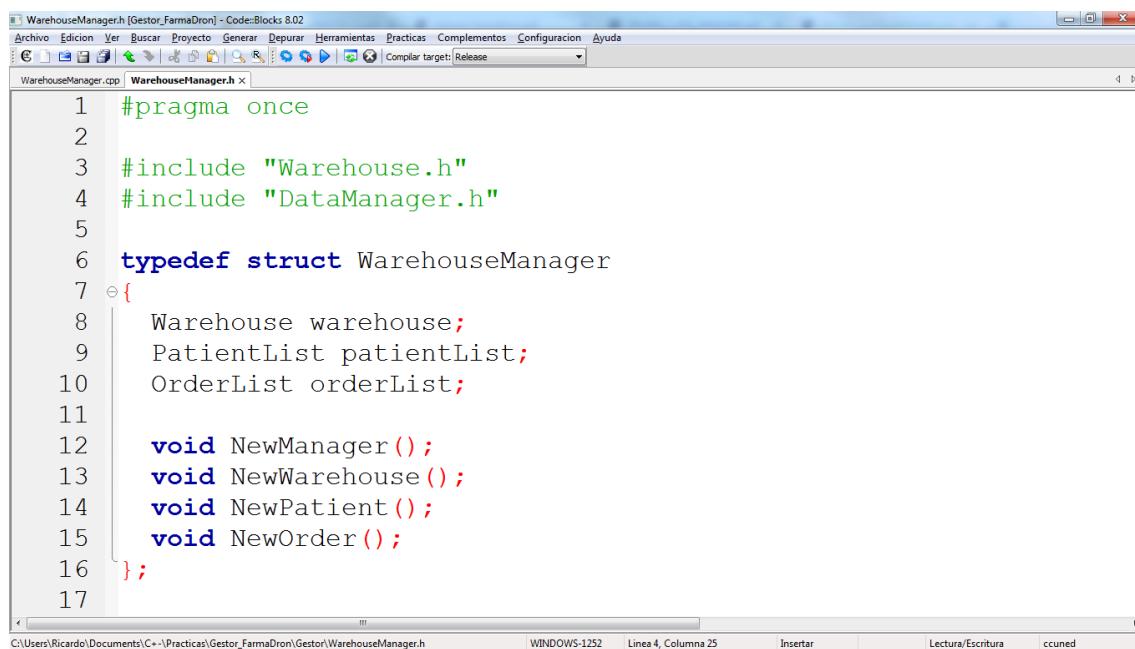
La aplicación trabajará con distintos almacenes; este módulo permite crear un tipo de dato que identifica a cada almacén según la información que maneja la aplicación y las funciones necesarias para poder operar con esa información.

Relación con otros módulos

Cualquier módulo que trabaje con los datos de los almacenes debe identificar el almacén que contiene esos datos utilizando el tipo de dato que define este módulo.

El módulo utiliza los módulos servicios.

Módulo gestor de almacén

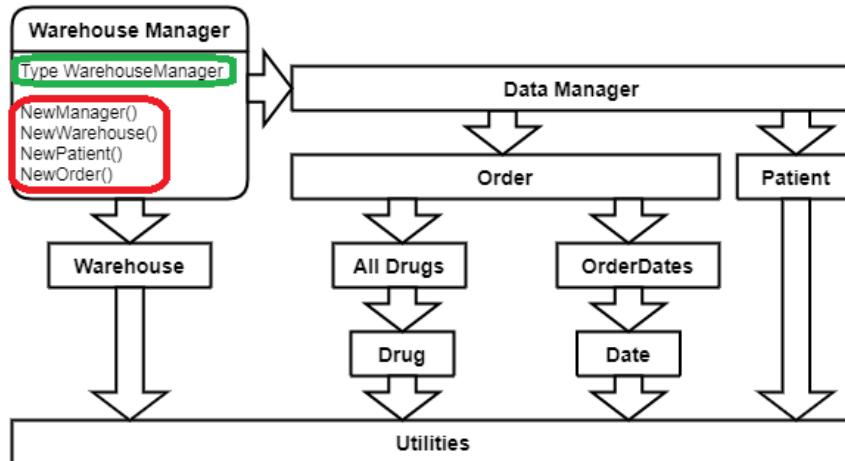


```
WarehouseManager.h [Gestor_FarmaDron] - Code:Blocks 8.02
Archivo Edición Ver Buscar Proyecto Generar Depurar Herramientas Prácticas Complementos Configuración Ayuda
C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\WarehouseManager.h Comilar target: Release
WarehouseManager.cpp WarehouseManager.h x
1 #pragma once
2
3 #include "Warehouse.h"
4 #include "DataManager.h"
5
6 typedef struct WarehouseManager
7 {
8     Warehouse warehouse;
9     PatientList patientList;
10    OrderList orderList;
11
12    void NewManager();
13    void NewWarehouse();
14    void NewPatient();
15    void NewOrder();
16 };
17
```

C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\WarehouseManager.h WINDOWS-1252 Línea 4, Columna 25 Insertar Lectura/Escritura ccuned

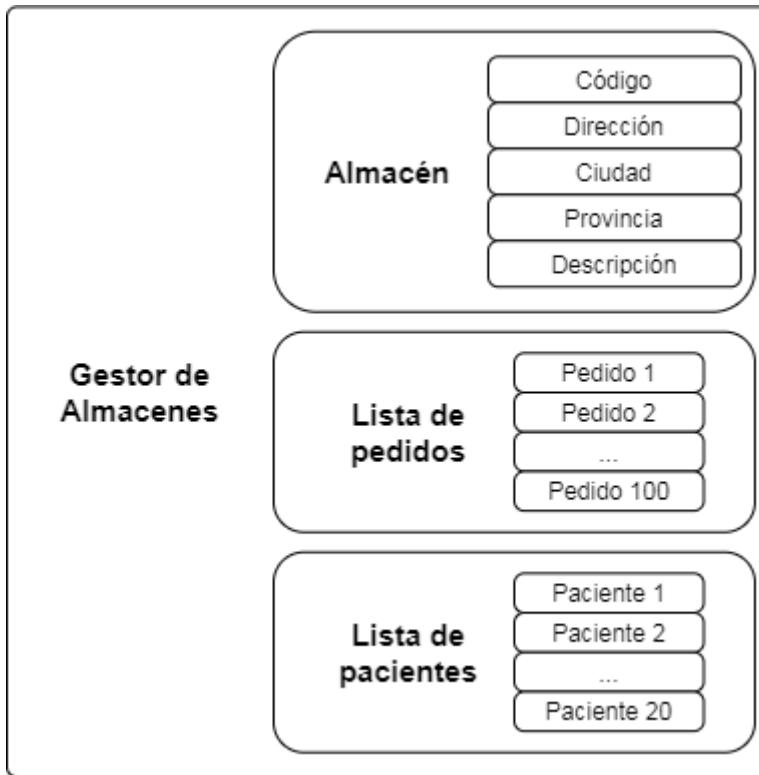
Elementos del módulo

Módulo Gestor de Almacenes



Tipo de dato

El tipo de dato *gestor de almacén* está formado por los datos: almacén, un array de pedidos y un array de pacientes se definieron en el módulo de gestión de datos del almacén.



- **Almacén** → Dato *Almacén*. Tipo de dato definido en el módulo almacén.
- **Lista de pedidos** → Array de datos *pedido*. Tipo de dato definido en el módulo gestor de datos almacén.
- **Lista de paciente** → Array de datos *paciente*. Tipo de dato definido en el módulo gestor de datos almacén.

Funciones

- **NewWarehouse**. Inicializa los datos del campo almacén de la variable del tipo *gestor de almacenes*.
- **NewPatient**. Inicializa uno de los datos del campo paciente de la lista de datos tipo *paciente* de la variable del tipo *gestor de almacenes*.
- **NewOrder**. Inicializa uno de los datos del campo pedido de la lista de datos tipo *pedido* de la variable del tipo *gestor de almacenes*.
- **NewManager**. Inicializa la variable del tipo *gestor de almacenes*.

Objeto del módulo

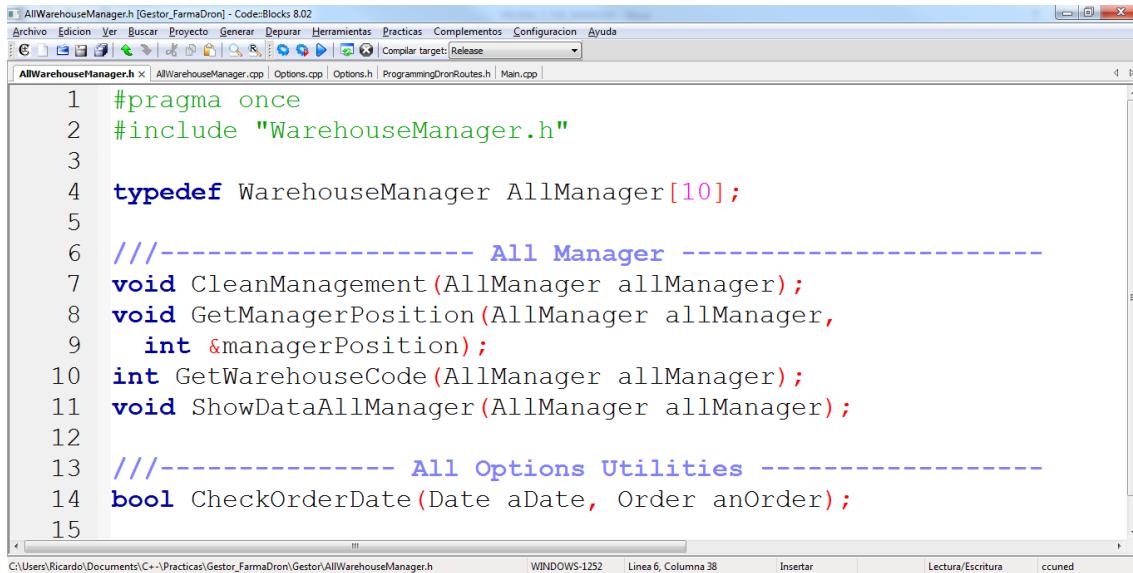
Crea un tipo de dato que relaciona los datos que identifican a un almacén con los datos de pedidos y pacientes que corresponden al almacén. Facilita las funciones que permiten operar con esa información.

Relación con otros módulos

Cualquier módulo que trabaje con los datos de los almacenes debe identificar el almacén que contiene esos datos utilizando el tipo de dato que define este módulo.

El módulo utiliza los módulos servicios, almacén, el paquete *almacén* y su módulo gestor.

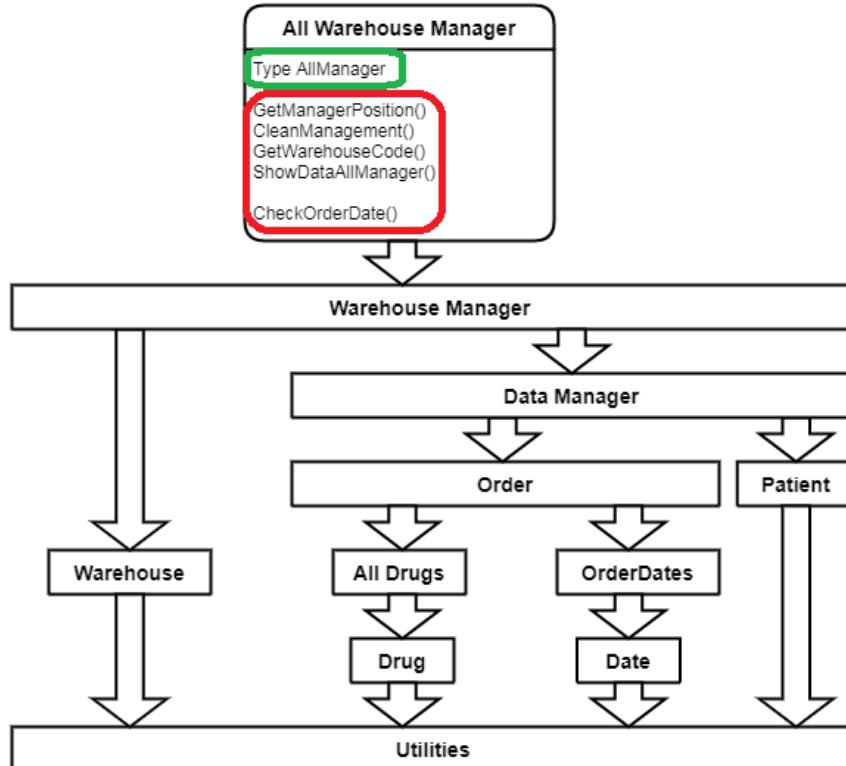
Módulo gestores de almacén



```
1 #pragma once
2 #include "WarehouseManager.h"
3
4 typedef WarehouseManager AllManager[10];
5
6 ----- All Manager -----
7 void CleanManagement(AllManager allManager);
8 void GetManagerPosition(AllManager allManager,
9     int &managerPosition);
10 int GetWarehouseCode(AllManager allManager);
11 void ShowDataAllManager(AllManager allManager);
12
13 ----- All Options Utilities -----
14 bool CheckOrderDate(Date aDate, Order anOrder);
15
```

Elementos del módulo

Módulo Gestores de Almacenes



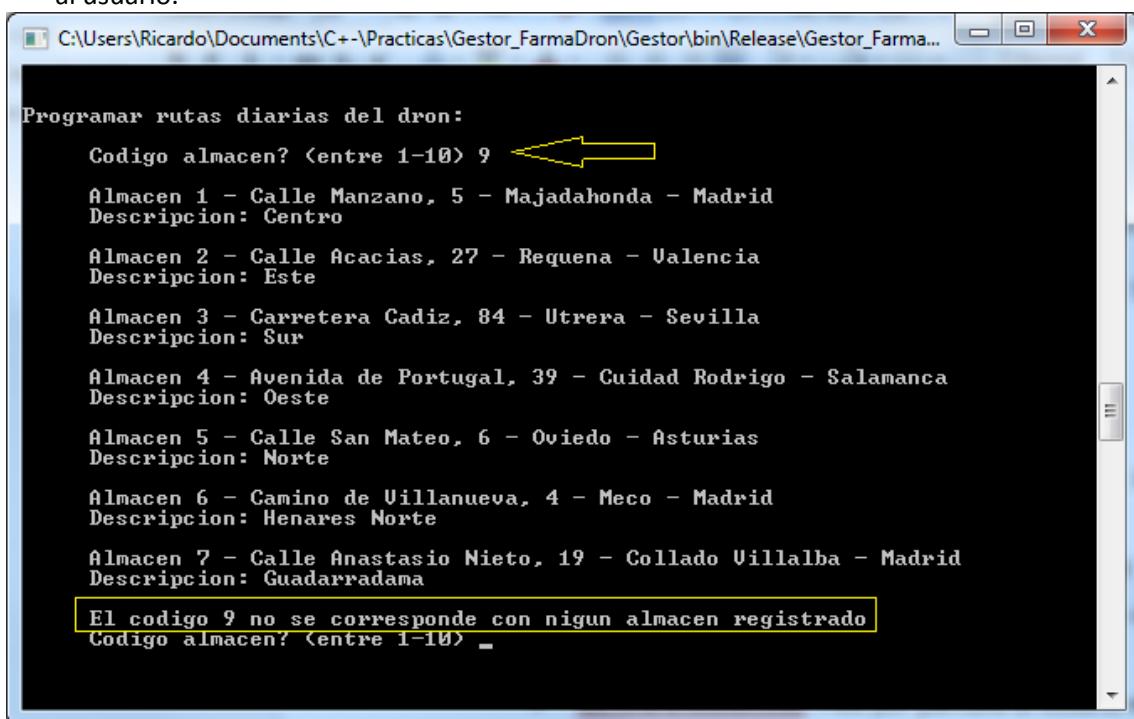
Tipo de dato

El tipo de dato *gestores de almacén* es un array de 10 datos de tipo *gestor de almacén*.

- **Gestor de almacén** → Dato *Gestor de almacén*. Tipo de dato definido en el módulo gestor de almacén.

Funciones

- **CleanManagement.** Resetea array resemando los datos que lo componen. Usa funciones definidas en los módulos anteriores.
- **GetManagerPosition.** Localiza la posición en el array de datos de tipo *gestor de almacenes* que ocupa el gestor del almacén con código que el usuario introduce por pantalla.
- **GetWarehouseCode.** Pide por pantalla al usuario el código de un almacén. Si el código del almacén dado por el usuario no se corresponde a ninguno registrado se lo comunica al usuario.



- **ShowDataAllManager.** La aplicación muestra por pantalla la información que tiene registrada hasta el momento.

```

C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
ListadoGestion FarmaDron

Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripción: Centro

Lista de pacientes y su ubicación:
Ref. Identificador Distancia Ángulo
1 María Carrera 5432 1990
2 Loreto Arenillas 7430 1255
3 Jesús Aguado 2965 54
4 Luis Miguel Brasero 3754 1934
5 Carolina Moreno 23 578

Pedidos:
Cliente Fecha Farmaco Peso Unidades
1 21/1/2021 Bicarbonato 2750 1
1 21/1/2021 Ibuprofeno 250 2
1 15/2/2021 Paracetamol 75 2
2 1/1/2021 Antihistamínico 100 1
2 30/1/2021 Omeprazol 350 2
2 20/2/2021 Ibuprofeno 50 3
3 1/1/2021 Alcohol 150 3
3 1/1/2021 Aspirina 50 2
3 10/1/2021 Omeprazol 750 1

Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripción: Este

Lista de pacientes y su ubicación:
Ref. Identificador Distancia Ángulo
1 José María Barrio 9034 1452
2 Eva Bailén 11789 378
3 Marta Llorente 334 250
4 Sergio Arribas 7113 994
5 Isabel Royo 643 12

Pedidos:
Cliente Fecha Farmaco Peso Unidades
1 30/3/2021 Ibuprofeno 25 2
1 30/3/2021 Bicarbonato 250 1

```

- **CheckOrderDate.** La función evalúa un pedido y una fecha e indica si el pedido será enviado al paciente en la fecha de la consulta.

Objeto del módulo

Crea un tipo de dato que centraliza toda la información que tiene registrada la aplicación de forma ordenada.

Proporciona las funciones que permiten trabajar con esa información.

Al ser un módulo al que recurrirán las funciones que trabajen con los datos registrados en la aplicación y no tener demasiado texto, en lugar de recurrir a un módulo separado para una función que usarán algunas funciones que desarrollan las opciones ofrecidas por la aplicación, la función **CheckOrderDate** se encuentra en él.

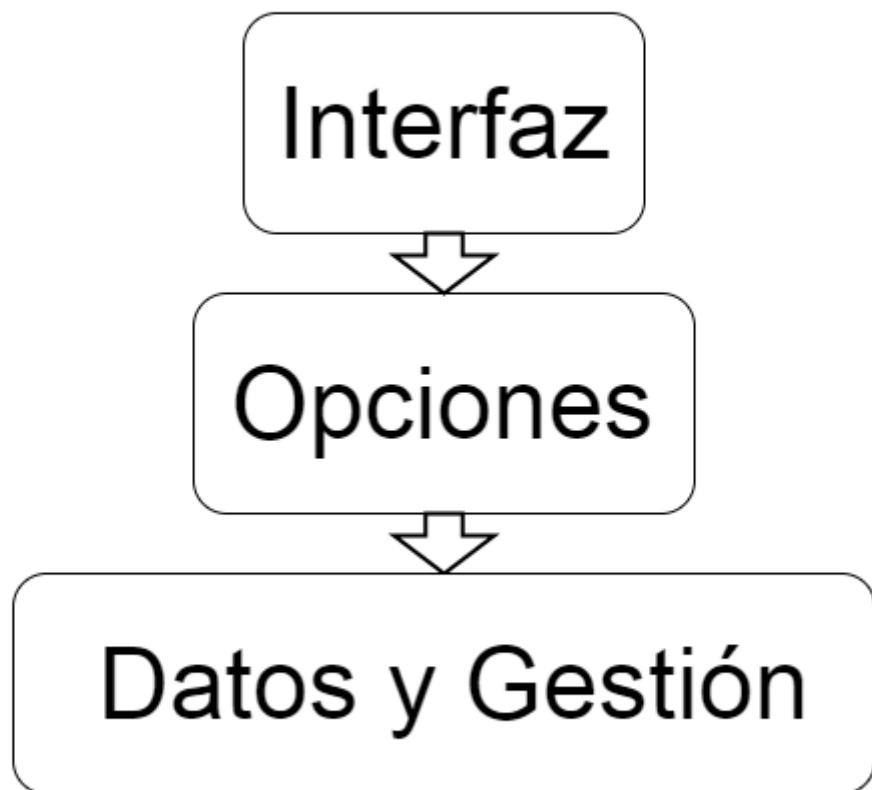
Relación con otros módulos

Cualquiera de las funciones de la aplicación debe trabajar con este módulo al ser el módulo que gestiona los datos registrados en la aplicación.

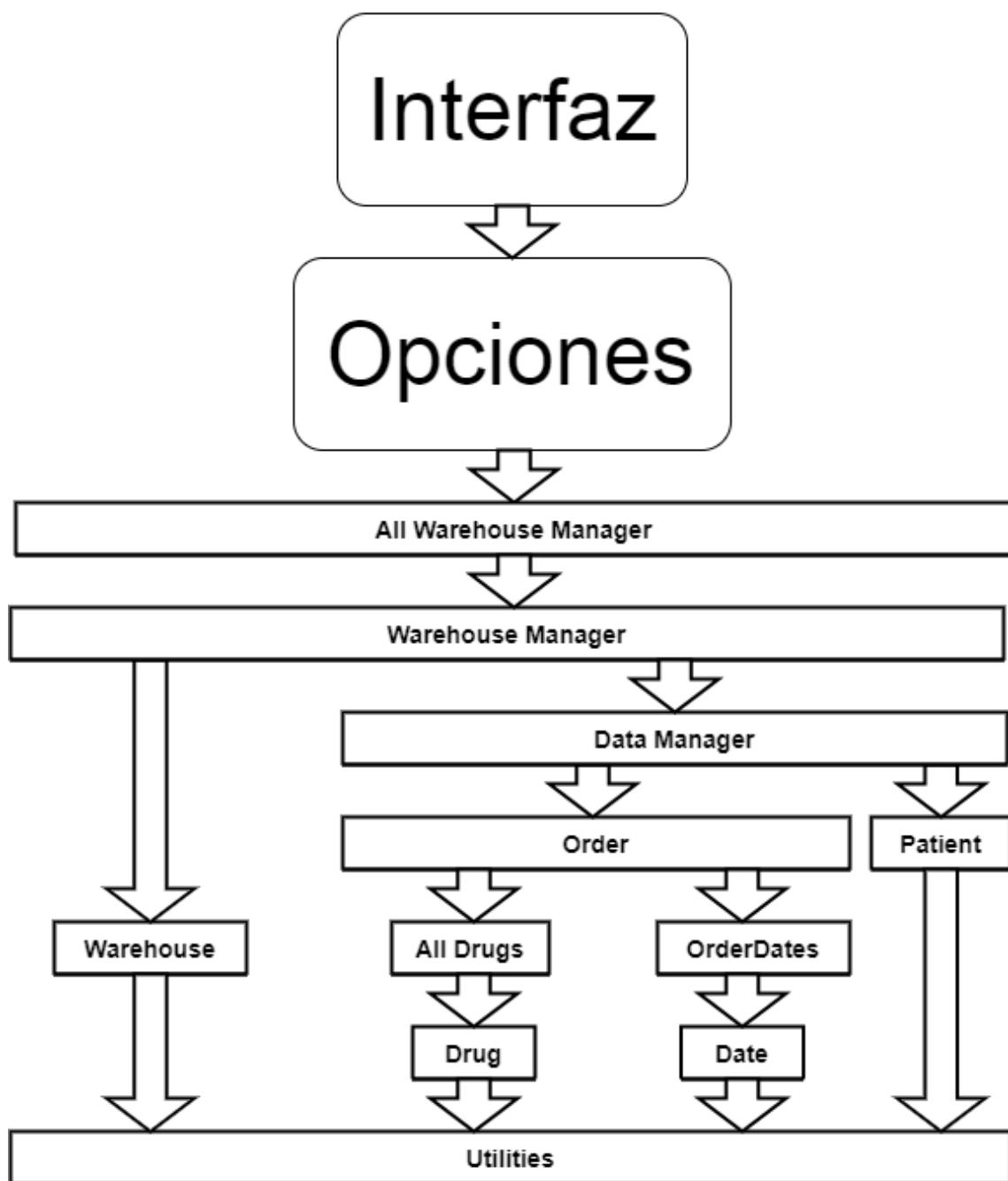
El módulo utiliza los módulos descritos hasta el momento.

Interfaz y opciones de la aplicación

Gestor FramaDron

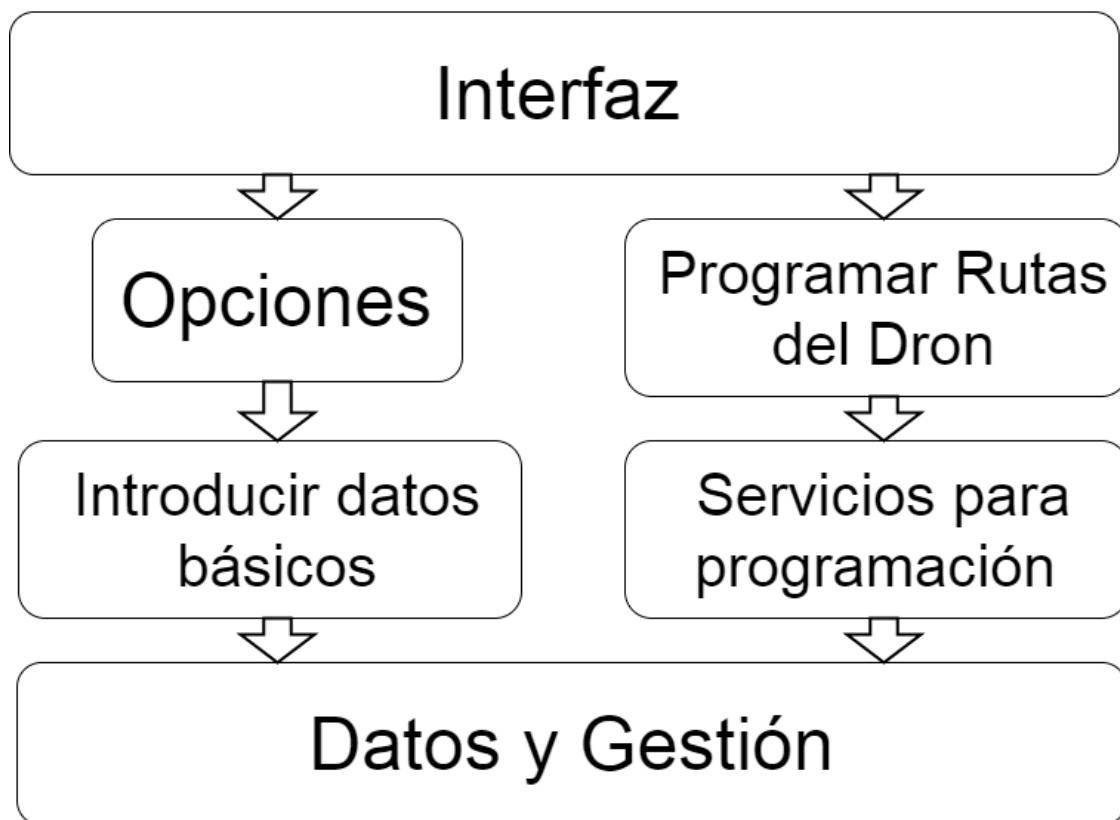


Gestor FramaDron



Opciones

Gestor FramaDron



Módulo introducir datos básicos

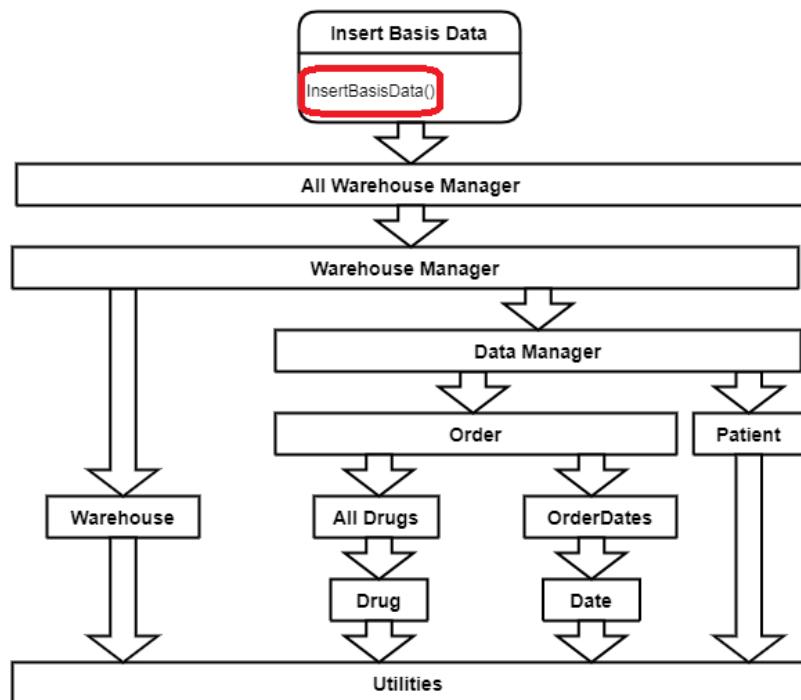
The screenshot shows the Code::Blocks IDE interface with the file 'InsertBasisData.h' open. The code in the editor is:

```
1 #pragma once
2 #include "AllWarehouseManager.h"
3
4 void InsertBasisData(AllManager allManager);
5
```

Below the editor, the status bar displays the path 'C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\InsertBasisData.h', the operating system 'WINDOWS-1252', line 'Linea 2, Columna 33', and the current mode 'Insertar Lectura/Escritura ccuned'.

Elementos del módulo

Módulo Introducir Datos Básicos



Tipo de datos

En este módulo no se define ningún nuevo tipo de dato.

Funciones

- **InsertBasisData.** Al ejecutarse la aplicación no tiene registrado ningún tipo de dato. Esta función contiene el registro de algunos almacenes pedidos y pacientes para poder comenzar a operar con información en la aplicación sin tener que registrarla antes, cólo

ejecutándola.

Objeto del módulo

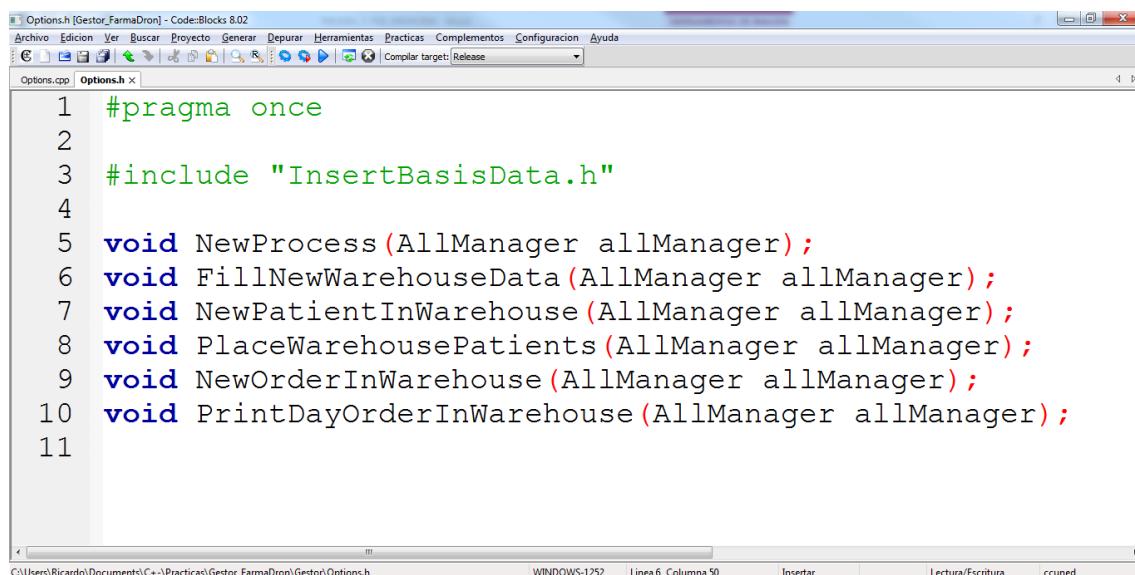
Es un módulo por completo subsidiario del módulo opciones; el único motivo para haberlo desgregado es poder trabajar más cómodamente sobre el código para manipular los datos de partida que si esos datos hubieran estado en el cuerpo del texto de un módulo más extenso con más funciones.

Relación con otros módulos

El módulo opciones ejecuta la función de este módulo para inicializar los datos básicos de la aplicación.

El módulo utiliza los módulos del paquete que se acaba de definir como datos y gestión.

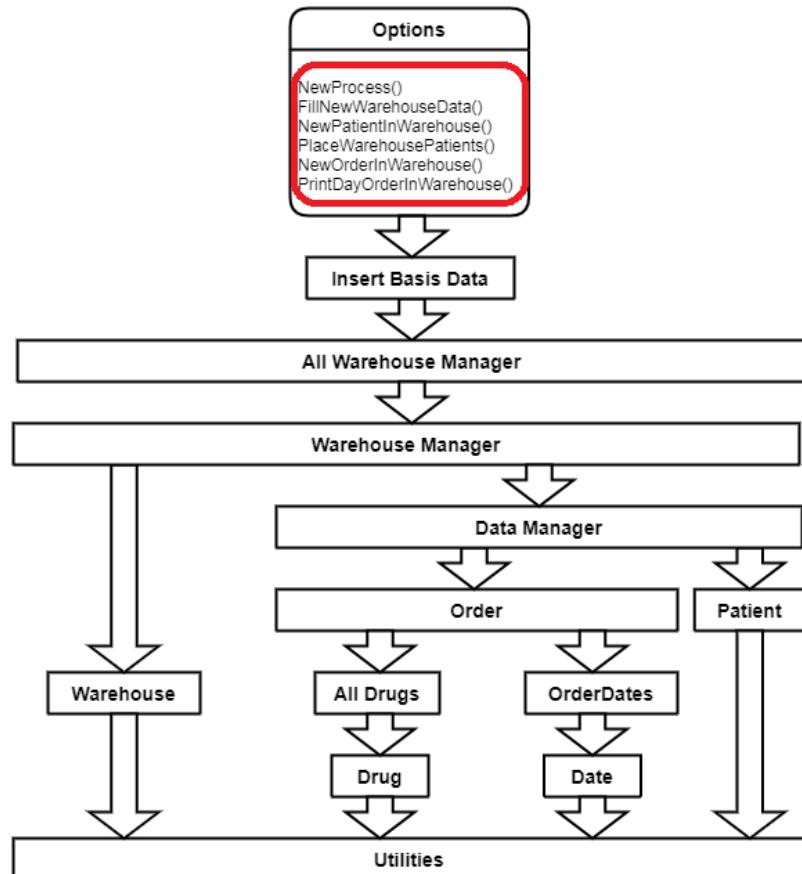
Módulo opciones



```
1 #pragma once
2
3 #include "InsertBasisData.h"
4
5 void NewProcess(AllManager allManager);
6 void FillNewWarehouseData(AllManager allManager);
7 void NewPatientInWarehouse(AllManager allManager);
8 void PlaceWarehousePatients(AllManager allManager);
9 void NewOrderInWarehouse(AllManager allManager);
10 void PrintDayOrderInWarehouse(AllManager allManager);
11
```

Elementos del módulo

Módulo Introducir Datos Básicos



Tipo de dato

En este módulo no se define ningún nuevo tipo de dato.

Funciones

- **NewProcess.** Inicializa la aplicación con la información mínima de arranque que se define en el módulo insertar datos básicos.
- **FillNewWarehouseData.** Permite dar de alta un nuevo almacén. Si todos los almacenes están ocupados, se preguntará al usuario por el almacén existente a eliminar.
- **NewPatientInWarehouse.** Permite dar de alta a un nuevo paciente en un almacén.
- **PlaceWarehousePatients.** Lista los pacientes y ubicaciones del almacén seleccionado.
- **NewOrderInWarehouse.** Permite realizar un nuevo pedido para un paciente de un almacén, puntual o periódico, de uno o varios fármacos.
- **PrintDayOrderInWarehouse.** Lista todos los pedidos de un almacén dado para un día concreto y las ubicaciones para su envío.

Objeto del módulo

El módulo agrupa todas las opciones que da la aplicación al usuario salvo la opción de programar las rutas del dron que por su extensión se disgrega en un módulo separado.

Relación con otros módulos

El módulo interfaz se comunica con este módulo para ejecutar las instrucciones del usuario.

El módulo utiliza su módulo auxiliar módulo introducir datos básicos y los módulos del paquete datos y gestión.

Módulo servicios para programación

The image displays three separate windows of the Code::Blocks IDE, each showing a portion of the same C++ header file, `ProgrammingUtilities.h`. The file is part of a project named "Gestor_FarmaDron".

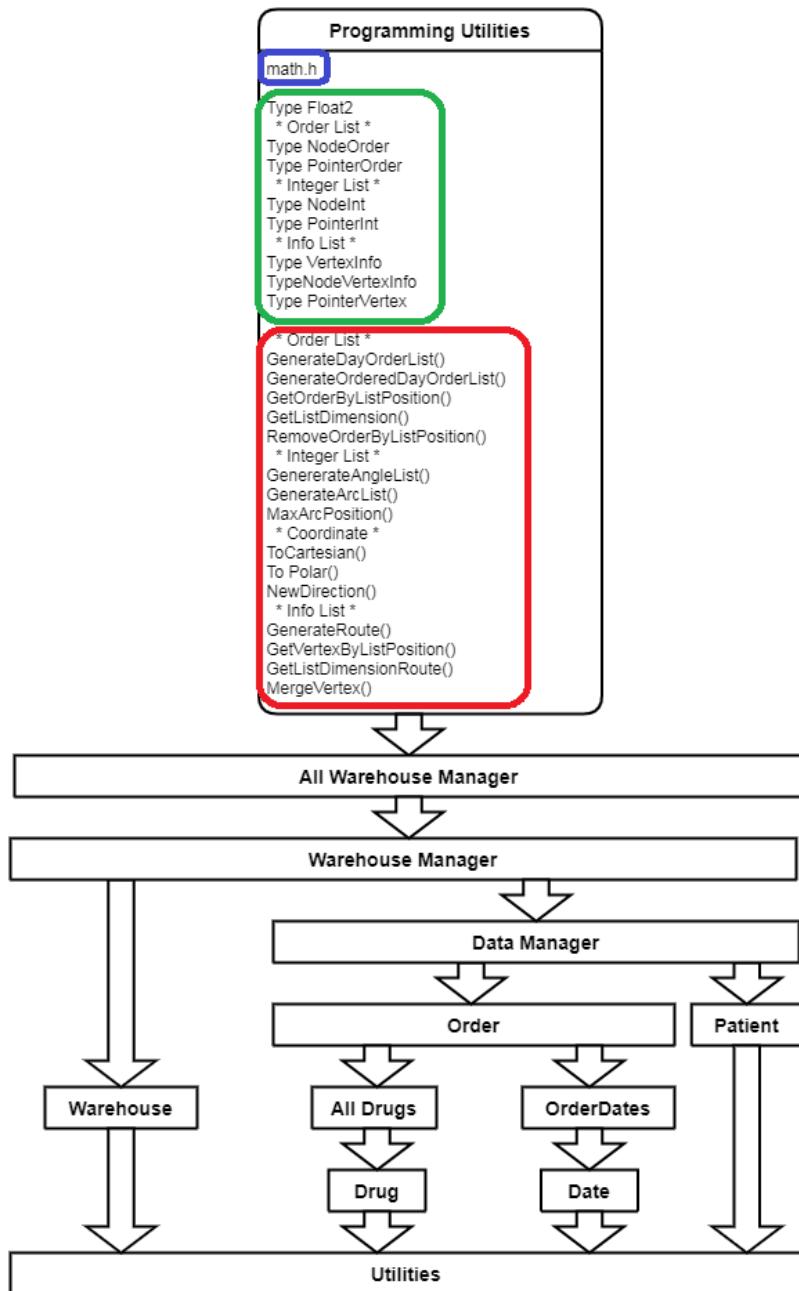
The code defines several utility functions and data structures:

- Order List:** A linked list of `NodeOrder` nodes. Each node contains a `Order dayOrder;` and a `NodeOrder* next;`.
- PointerOrder:** A pointer to a `NodeOrder` node.
- GenerateDayOrdersList:** A function that generates a list of day orders.
- GenerateOrderedDayOrdersList:** A function that generates an ordered list of day orders.
- GetOrderByListPosition:** A function that retrieves an order by its position in the list.
- GetListDimension:** A function that returns the dimension of the list.
- RemoveOrderByListPosition:** A function that removes an order from the list at a specific position.
- Integer List:** A linked list of `NodeInt` nodes. Each node contains an `int integer;` and a `NodeInt* next;`.
- PointerInt:** A pointer to a `NodeInt` node.
- GenerateAngleList:** A function that generates a list of angles.
- GenerateArcList:** A function that generates a list of arcs.
- MaxArcPosition:** A function that finds the maximum arc position.
- Coordinate Functions:** Functions to convert between Cartesian and Polar coordinates.
- Info List:** A linked list of `NodeInt` nodes, identical in structure to the Integer List.

The code is heavily annotated with comments indicating the purpose of each section and function.

Elementos del módulo

Módulo Servicios para Programación



Tipos de dato

En este módulo se definen listas de tres tipos de datos:

- Lista de datos tipo *pedido*.
- Lista de números enteros.
- Lista de Información sobre los pedidos. El dato *información sobre pedido* será un tipo de dato que también se define en el módulo.

Para la creación de la lista de datos *pedido* se crean los datos:

- Nodo *pedido*.
- Puntero *pedido*.

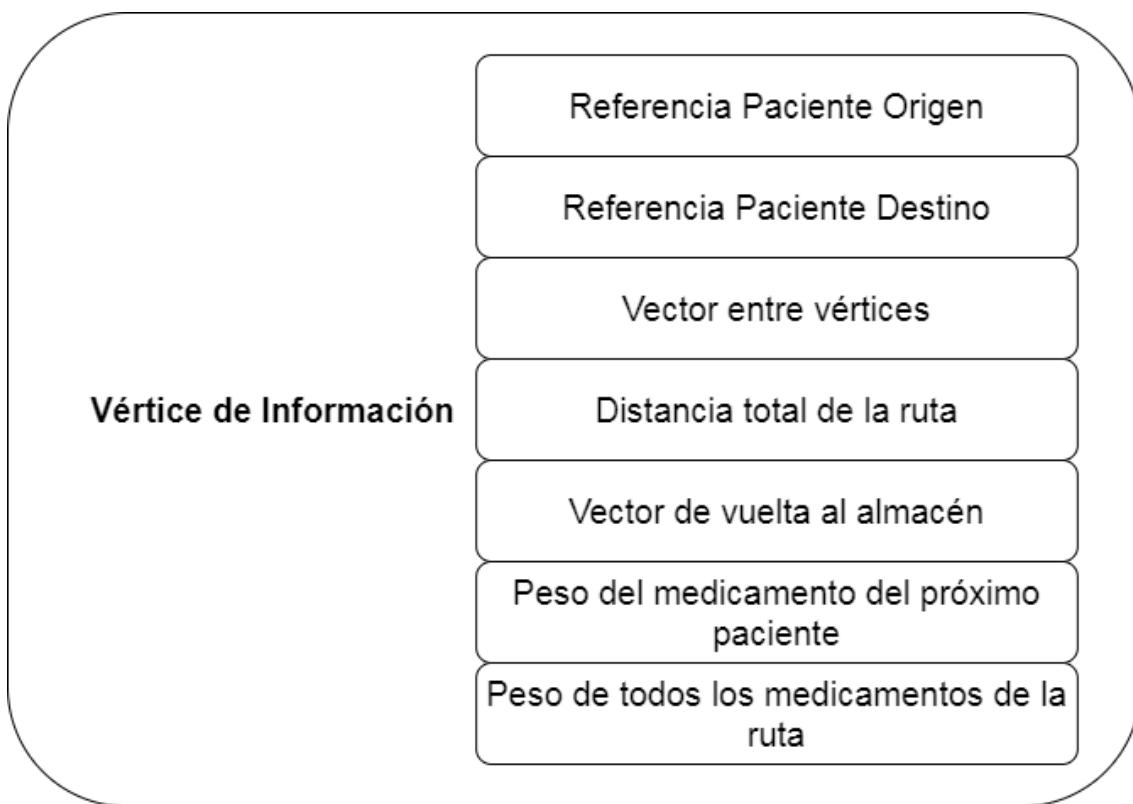
Para la creación de las listas de números enteros se crean los datos:

- Nodo número entero.
- Puntero número entero.

Para la creación de la lista de datos *información sobre pedido* se crean los datos:

- Vértice información.
- Nodo vértice de información.
- Puntero vértice de información.

El tipo de dato vértice de información, definido en este módulo, está compuesto por cinco campos.



- **Referencia Paciente Origen** → Número entero. Tipo de dato reconocido por C+-.
- **Referencia Paciente Destino** → Número entero. Tipo de dato reconocido por C+-.
- **Vector entre vértices** → Array de número reales. Tipo de dato definido en este módulo.
- **Distancia total de la ruta** → Número real. Tipo de dato reconocido por C+-.
- **Vector de vuelta al almacén** → Array de número reales. Tipo de dato definido en este módulo.
- **Peso del medicamento del próximo paciente** → Número entero. Tipo de dato reconocido por C+-.

- **Peso de todos los medicamentos de la ruta** → Número entero. Tipo de dato reconocido por C++.

Funciones

Lista de datos de tipo *pedido*.

- **GenerateDayOrdersList**. Genera una lista de datos tipo *pedido* con los pedidos que se le van pasando por parámetro.
- **GenerateOrderedDayOrdersList**. Genera una lista ordenada según el ángulo de la posición del paciente de cada pedido de datos tipo *pedido* con los pedidos que se le van pasando por parámetro.
- **GetOrderByListPosition**. Devuelve el dato tipo *pedido* que ocupa la posición que se le pasa por parámetro de la lista de datos tipo *pedido* que se le pasa por parámetro.
- **GetListDimension**. Devuelve la dimensión de la lista, el número de datos tipo *pedido*, de la lista que se le pasa por parámetro.
- **RemoveOrderByListPosition**. Elimina el dato tipo *pedido* de la lista de datos que se la pasa por parámetro que ocupa la posición que se le pasa por parámetro.

Lista de números enteros.

- **GenerateAngleList**. Genera una lista de números enteros con los números enteros que se le van pasando por parámetro.
- **GenerateArcList**. Genera una lista de números enteros que se corresponden con la diferencia que hay entre los valores de la lista de números enteros que se le pasa por parámetro.
- **MaxArcPosition**. Devuelve la posición en la lista del mayor número entero.

Coordenadas

- **ToCartesian**. Devuelve en coordenadas cartesianas el vector en coordenadas polares que se le pasa por parámetro.
- **ToPolar**. Devuelve en coordenadas polares el vector en coordenadas cartesianas que se le pasa por parámetro.
- **NewDirection**. Devuelve en la variable del primer parámetro el vector en coordenadas polares que une los dos puntos que se pasan por parámetro.

Lista de datos *información sobre pedido*

- **GenerateRoute**. Genera una lista de datos tipo *información sobre pedido* con los datos *información sobre pedido* que se le van pasando por parámetro.

- **GetVertexByListPosition.** Devuelve el dato tipo *información sobre pedido* que ocupa la posición que se le pasa por parámetro de la lista de datos tipo *información sobre pedido* que se le pasa por parámetro.
- **GetListDimensionRoute.** Devuelve la dimensión de la lista, el número de datos tipo *información sobre pedido*, de la lista que se le pasa por parámetro.
- **MergeVertex.** Analiza los datos *información sobre pedido* de la lista que se le pasa por parámetro y los trata para darles un sentido en el contexto de la aplicación.

Objeto del módulo

Módulo auxiliar del módulo que calculará las rutas diarias del dron en el que se agrupan las funciones principales y los nuevos datos para el algoritmo que creará las rutas.

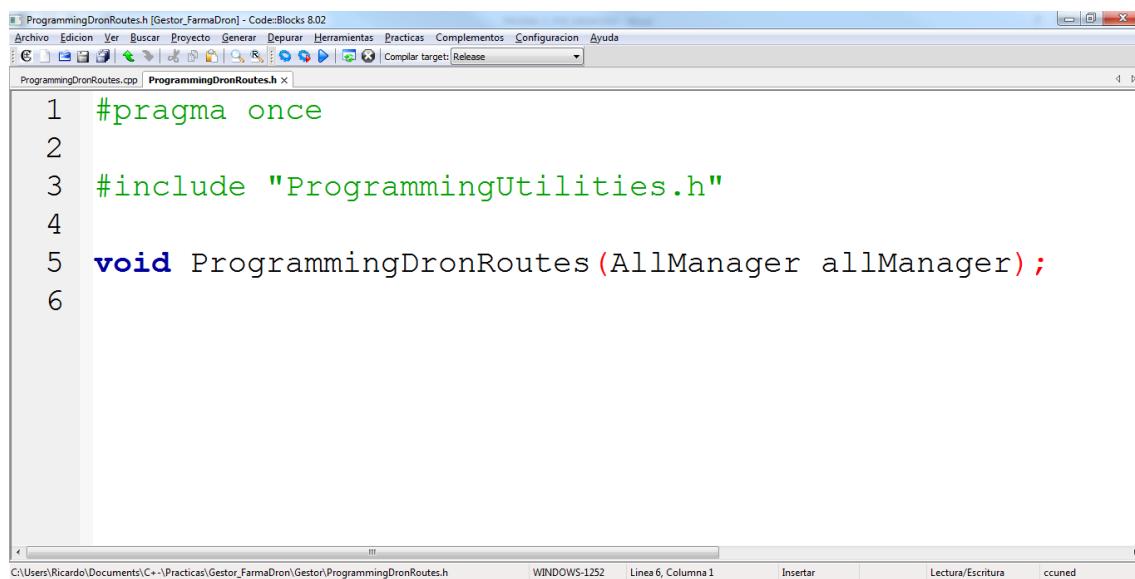
El objetivo del módulo es aligerar y simplificar el módulo programar rutas del dron.

Relación con otros módulos

El módulo programar rutas del dron usa como apoyo imprescindible este módulo.

El módulo utiliza los módulos del paquete de módulos que se define como datos y gestión.

Módulo programar rutas del dron

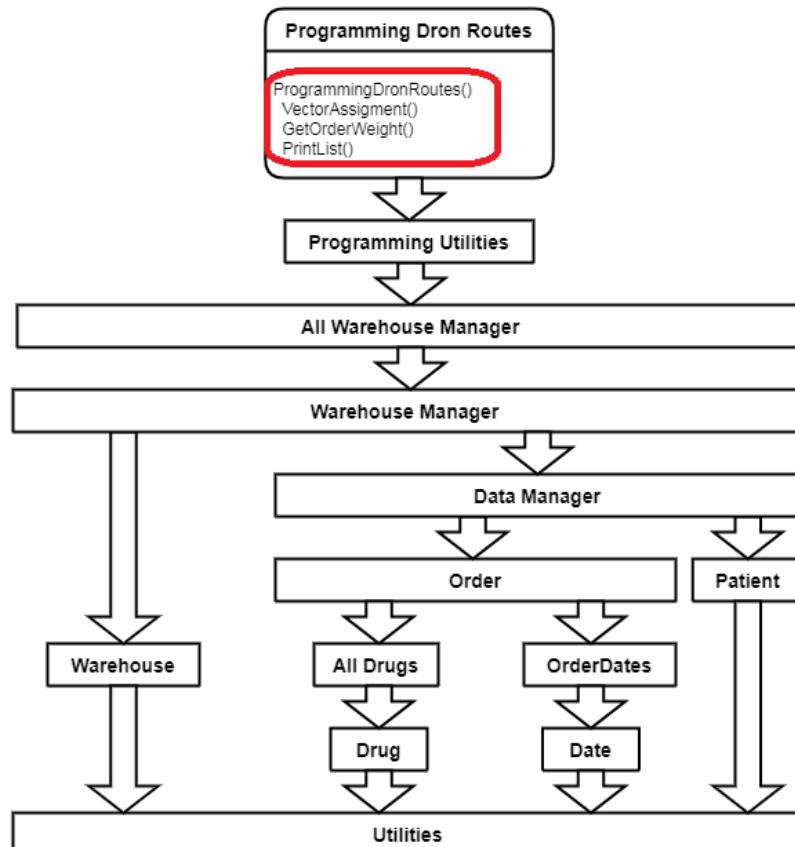


```
ProgrammingDronRoutes.h [Gestor_FarmaDron] - Code::Blocks 8.02
Archivo Edición Ver Buscar Proyecto Generar Depurar Herramientas Prácticas Complementos Configuración Ayuda
C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\ProgrammingDronRoutes.h | Comilar target: Release |
ProgrammingDronRoutes.cpp | ProgrammingDronRoutes.h | x
1 #pragma once
2
3 #include "ProgrammingUtilities.h"
4
5 void ProgrammingDronRoutes(AllManager allManager);
```

C:\Users\Ricardo\Documents\C++\Prácticas\Gestor_FarmaDron\Gestor\ProgrammingDronRoutes.h WINDOWS-1252 Línea 6, Columna 1 Insertar Lectura/Escritura ccuned

Elementos del módulo

Módulo Prógramar Rutas del Dron



Tipo de dato

En este módulo no se define ningún nuevo tipo de dato.

Funciones

- **VectorAssignment.** Asigna a un array los valores de otro.
- **GetOrderWeight.** Devuelve el peso del pedido que se le pasa por parámetro.
- **PrintList.** Muestra por pantalla información sobre la ruta del dron con un determinado formato.

```
C:\Users\Ricardo\Documents\C+->\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Código almacén? <entre 1-10> 2
Dia? 30
Mes? 3
Año? 2021

Ruta 1
Origen a Cliente1 -- Distancia recorrida: 9034 Angulo: 1452 Peso: 1150
Cliente1 a Origen -- Distancia recorrida: 9034 Angulo: 452 Peso: 0
Distancia total de la ruta: 18068

Ruta 2
Origen a Cliente2 -- Distancia recorrida: 11789 Angulo: 378 Peso: 230
Cliente2 a Origen -- Distancia recorrida: 11789 Angulo: 1378 Peso: 0
Distancia total de la ruta: 23578

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

- **ProgrammingDronRoutes.** Pregunta por pantalla al usuaria el código de un alamacen y una fecha y le devuelve las rutas que seguirá el dron para hacer el reparto.

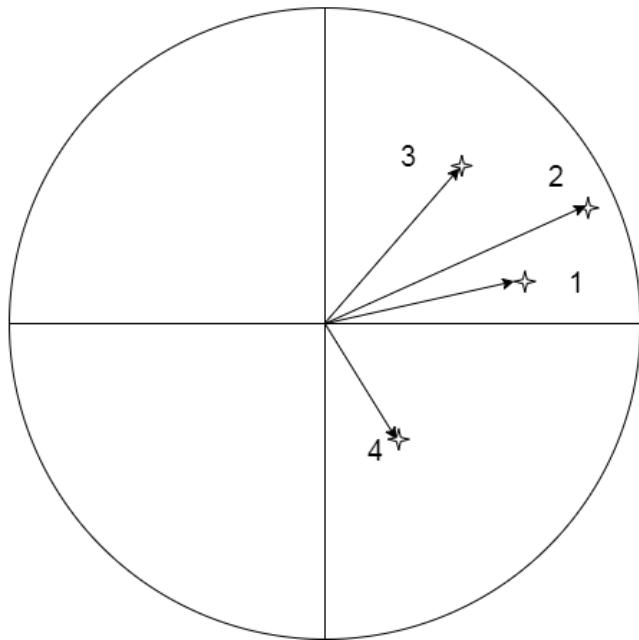
El algortimo que sigue el método es el siguiente:

1.- Configuración.

Pide al ususario el código del almacén y la fecha de las que necesita saber las rutas del dron.

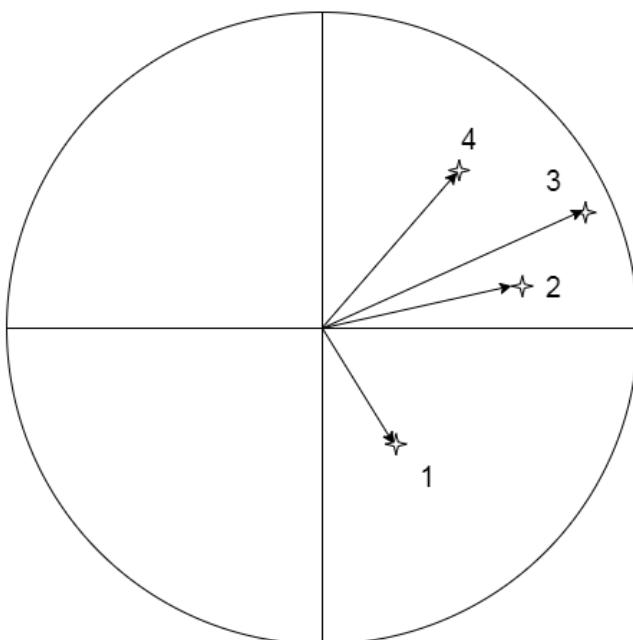
2.- Ordena los pedidos de la fecha solicitada por el usuario según el ángulo de la localización del paciente de menor a mayor.

Orden según el ángulo de la localización del paciente



3.- Calcula los ángulos que hay entre las distintas localizaciones de los pacientes, toma como referencia los dos pacientes que se encuentran separados por un ángulo mayor y reordena la lista tomando a los dos pacientes a mayor separación como primero y último en sentido antihorario.

Orden según las separaciones entre las localizaciones de los pacientes



5.- Siguiendo el orden del punto anterior, calcula la información de cada pedido que resulta relevante en el diseño de las rutas del dron: peso del pedido y distancia a recorrer.

6.- En el orden proporcionado por la lista anterior, se evalúa si el dron puede viajar a la localización del siguiente pedido o debe volver al almacén.

Con las localizaciones que están dentro del radio de acción del dron, se genera la ruta.

7.- Los pedidos de la ruta anterior se retiran de la lista de pedidos del día y se vuelven a evaluar las localizaciones de los pedidos generando nuevas rutas.

El proceso se repite hasta que todos los pedidos del día se encuentran en alguna ruta del dron.

Objeto del módulo

La función se separa del resto del módulo opciones por ser más compleja y extensa que el resto.

Relación con otros módulos

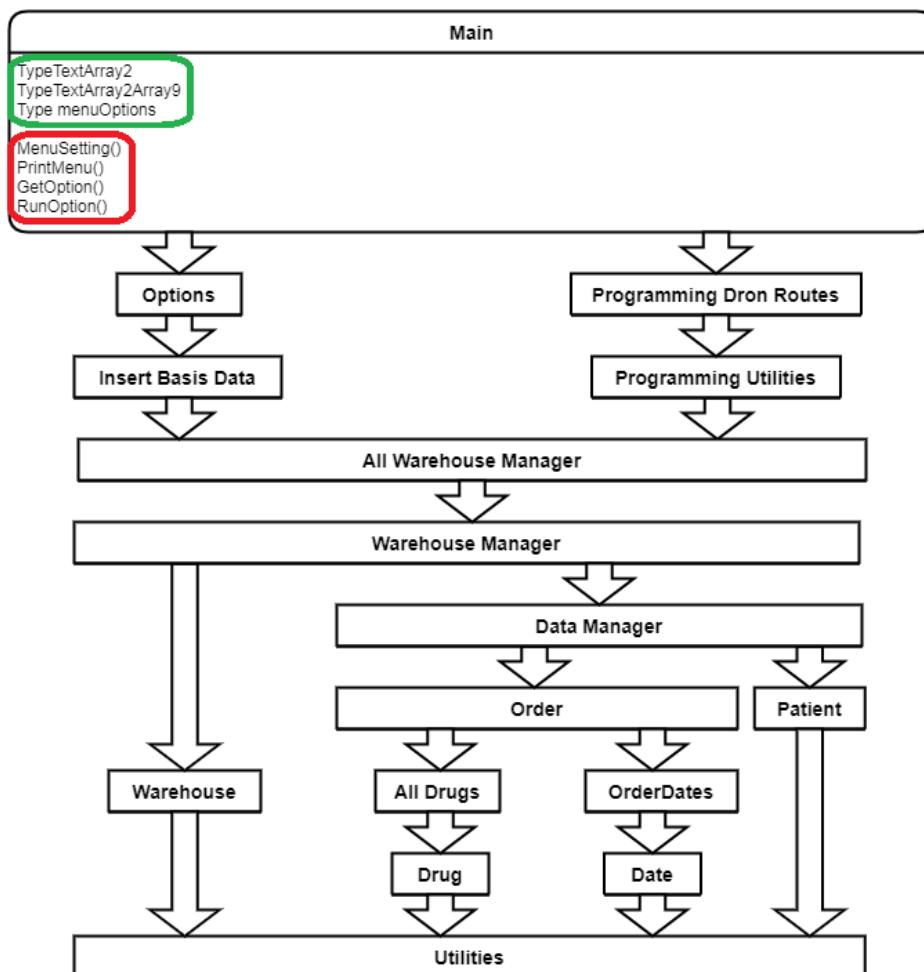
La función programar rutas del dron es ejecutada por el módulo interfaz.

El módulo utiliza los módulos del paquete de módulos que se define como datos y gestión y el módulo servicios de programación que es un modo auxiliar de éste.

Módulo interfaz

Elementos del módulo

Módulo Interfaz



Tipos de datos

Los datos tipo *TextArea2* y *TextArea2Array9* son tipos de datos auxiliares que se crearon para imprimir por pantalla el menú de las opciones de la aplicación. No se definen en el módulo servicios porque sólo se usarán en una función de módulo interfaz.

El dato tipo *MenuOptions* es un dato de tipo enumerado.

```
13 typedef enum menuOptions
14 {
15     newProcess,           // 0
16     newWarehouse,         // 1
17     newPatientInWarehouse, // 2
18     patientLocation,      // 3
19     newOrder,             // 4
20     orderDay,             // 5
21     dronRoutesProgramming, // 6
22     dronRoutesRepresent,   // 7
23     quit,                 // 8
24     none                  // 9
25 };
26
27 menuOptions optionSelected = menuOptions(9);
```

Funciones

- **MenuSetting.** Inicializa algunas variables necesarias para imprimir por pantalla el menú de las opciones de la aplicación.
- **PrintMenu.** Imprime por pantalla el menú con las opciones de la aplicación.
- **GetOption.** Recibe la información del usuario sobre la opción que se ejecutará.
- **RunOption.** Ejecuta la opción seleccionada por el usuario.

Objeto del módulo

El módulo que realmente ejecuta la aplicación y establece la comunicación entre el usuario y la información que almacena la aplicación.

Relación con otros módulos

Se relaciona con todos los demás módulos de la aplicación.

2.- Posibles mejoras.

Funciones que se puedan incluir.

En el tratamiento de los datos de la práctica la información se registra y se expone, pero no se edita; si un dato necesita ser variado la única posibilidad que se ofrece al usuario es borrar el dato y volver a registrarla con las variaciones que necesite introducir. Este proceso puede convertir el manejo de la aplicación por parte del usuario, sobre todo cuando se tenga un amplio volumen de datos registrados, en una tarea tediosa que con el tiempo pueda llevar al abandono de uso de la aplicación.

Funciones para editar cualquier tipo de dato serían el siguiente paso en la evolución de la aplicación.

Mejora algorítmicas

La optimización del diseño de rutas del dron requiere un estudio de matemática discreta; buscar un ciclo Hamiltoniano que pondere las aristas según las restricciones del dron.

La evolución del algoritmo aportado en la resolución de la práctica sería un algoritmo que no siguiera un orden determinado visitando los pacientes, como hace el algoritmo actual con el sentido antihorario, sino que según los peso de los pedidos y las distancias a las que se encuentre el paciente ponderara los recorridos creando el orden de visita de las localizaciones que minimizara el recorrido según esa ponderación.

Proponer mejoras para la gestión de almacenes requiere conocer la gestión de almacenes, lo que dificulta cualquier propuesta de optimización de la aplicación sin tener un usuario real.

Partiendo de la premisa anterior y como las optimizaciones tienden a ahorrar recursos una posible optimización sería estudiar las localizaciones de los almacenes de forma que si dos se encuentran próximos y ninguno está cerca de alcanzar sus límites operativos, se centralizara la actividad del dron en uno y se cerrara el otro reduciendo así los costes de mantener ambos abiertos. De esta forma y siguiendo esa línea se podría establecer algún tipo de proceso que estudiara la actividad de cada uno haciendo redistribuciones de la carga de trabajo según las distintas situaciones operativas en que se encuentren.

Estructura de datos

A la hora de estructurar los datos en la aplicación se ha seguido una construcción modular de cada tipo de dato, estructuras de datos pequeñas que permitieran crear tipo de datos funcionales según las necesidades hasta al final llegar a un tipo de dato que los estaba conteniendo a todos y tenía fácil acceso a todos.

La evolución en los datos no sería tanto en cuanto a su estructura si no en crear listas dinámicas en lugar de arrays incluso en los datos que tienen un límite fijo; por ejemplo en una nueva versión de esta aplicación se usarían listas dinámicas en lugar de arrays para almacenar datos de tipo *paciente* o *pedido* imponiéndose en las listas los límites de elementos que exigiese la aplicación.

3.- Pruebas realizadas con el programa resuelto.

Listado de pruebas realizadas.

- 1.- Prueba de Inicialización de del sistema.
- 2.- Prueba de alta de almacén.
- 3.- Prueba de borrado almacén.
- 4.- Alta nuevo paciente en un almacén sin pacientes.
- 5.- Alta de nuevos pacientes en distintos almacenes.
- 6.- Registro de pedido en un paciente no registrado en el almacén.
- 7.- Registro de un pedio puntual de un único medicamento.
- 8.- Registro de un pedio puntual de un único medicamento una fecha que ya tiene medicamentos registrados.

- 9.- Registro de un pedido periódico.
- 10.- Programar ruta del dron con un solo pedido.
- 11.- Programar ruta del dron. Exceso de peso en la ruta.
- 12.- Programar ruta del dron. Vuelta al origen por falta de autonomía y pedido hecho sin necesidad de volver al almacén.
- 13.- Programar ruta del dron. Fusión de pedidos en un único viaje.
- 14.- Programar ruta del dron. Fusión de algunos pedidos en un único viaje.

Pruebas.

1.- Prueba de inicialización del sistema.

Descripción de la prueba

La aplicación inicializará con información mínima de arranque y la mostrará por pantalla los valores con que se inicializó y nada más. No expondrá ningún valor que contengan las variables que no se corresponda con la información mínima de arranque.

Entrada de datos

2 Almacenes:

1.- Majadahonda; Calle Manzano, 5; Madrid; Centro.

2.- Requena; Calle Acacias, 27; Valencia; Este.

1 Paciente:

1.- María; Carrera; 5432; 1990.

1 Pedido:

Referencia de paciente: 1.

Número de envíos: 1.

Fecha del envío: 21 de enero de 2021.

Medicamento: Bicarbonato.

Peso: 2750.

Unidades: 1.

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes           <Pulsar U>
    Nuevo pedido               <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I|M|A|U|N|L|P|R|S>?
I

ListadoGestion FarmaDron

Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripcion: Centro

Lista de pacientes y su ubicacion:
Ref. Identificador      Distancia     Angulo
1     Maria Carrera       5432          1990

Pedidos:
Cliente   Fecha        Farmaco      Peso   Unidades
1         21/1/2021    Bicarbonato  2750      1

Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripcion: Este

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

2.- Prueba de Alta de almacén.

Descripción de la prueba

Se da de alta un almacén introduciendo los datos de pantalla; se comprueba que las cadenas de texto se registran correctamente y que la muestra de datos por pantalla tiene el formato correcto.

Entrada de datos

3; Avenida de la prueba, 1; Cuidad Ficticia; Provincia; Éste es un comentario de prueba para un almacén de prueba.

Capturas de salida

Para mostrar los almacenes registrados en la aplicación, introduzco el código de un almacén inexistente; cuando el usuario quiere registrar un paciente en un almacén con un código en el que no hay ningún almacén registrado, la aplicación muestra los almacenes que tiene registrados para que el usuario pueda comprobar a que se debe el error.

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Codigo almacen? <entre 1-10> 9
Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripcion: Centro
Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripcion: Este
El codigo 9 no se corresponde con ningun almacen registrado
Codigo almacen? <entre 1-10> _
```

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma... X

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes          <Pulsar U>
    Nuevo pedido               <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I:M:A:U:N:L:P:R:S>?
M

Alta nuevo almacen:
    Identificador almacen <cod. de alamacen 1 a 10>? 3
    Direccion almacen? <entre 1 y 48 caracteres> Avenida de la prueba, 1
    Municipio almacen? <entre 1 y 48 caracteres> Ciudad Ficticia
    Provincia almacen? <entre 1 y 16 caracteres> Provincia
    Descripcion almacen? <entre 1 y 48 caracteres> Este es un comentario de prueba para un almacen de prueba
    Datos correctos <S/N>? S

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes          <Pulsar U>
    Nuevo pedido               <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I:M:A:U:N:L:P:R:S>?
U

Lista de pacientes y su ubicacion:
    Codigo almacen? <entre 1-10> 9
    Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
    Descripcion: Centro
    Almacen 2 - Calle Acacias, 27 - Requena - Valencia
    Descripcion: Este
    Almacen 3 - Avenida de la prueba, 1 - Ciudad Ficticia - Provincia
    Descripcion: Este es un comentario de prueba para un almacen de prueba
    El codigo 9 no se corresponde con ningun almacen registrado
    Codigo almacen? <entre 1-10> _
```

3.- Prueba de borrado de almacén.

Descripción de la prueba

Se completa la capacidad de registro para almacenes de la aplicación. Se registra un nuevo almacén obligando a la aplicación a que el usuario borre uno de los almacenes ya registrados. Se comprueba que no quedan datos del antiguo almacén aunque las cadenas de texto del almacén borrado fueran más largas que las cadenas de texto del nuevo almacén.

Entrada de datos

```
C:\Users\Ricardo\Documents\C++\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
El sistema esta completo
Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripcion: Centro

Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripcion: Este

Almacen 3 - Carretera Cadiz, 84 - Utrera - Sevilla
Descripcion: Sur

Almacen 4 - Avenida de Portugal, 39 - Cuidad Rodrigo - Salamanca
Descripcion: Oeste

Almacen 5 - Calle San Mateo, 6 - Oviedo - Asturias
Descripcion: Norte

Almacen 6 - Calle de las flores, 4 - Priego - Cuenca
Descripcion: Serrania

Almacen 7 - Calle Rodrigo Diaz, 15 - Villarcayo - Burgos
Descripcion: Las Merindades

Almacen 8 - Calle Moron, 21 - Pastrana - Guadalajara
Descripcion: Alcarria

Almacen 9 - Avenida Iniesta, 3 - Plasencia - Caceres
Descripcion: Extremadura

Almacen 10 - Avenida de la amistad y la libertad, 1234 - Arenas de San Pedro
o - Avila
Descripcion: Este es un comentario muy largo para hacer pruebas

Introduzca el codigo del almacen que eliminar del sistema: 10

El almacen con codigo: 10 se borro del sistema
Alta nuevo almacen:

Identificador almacen <cod. de almacen 1 a 10>? 10
Direccion almacen? <entre 1 y 48 caracteres> Calle Sol, 1
Municipio almacen? <entre 1 y 48 caracteres> Ibi
Provincia almacen? <entre 1 y 16 caracteres> Alicante
Descripcion almacen? <entre 1 y 48 caracteres> Alm Alc

Datos correctos <S/N>? S
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C++\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripcion: Centro

Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripcion: Este

Almacen 3 - Carretera Cadiz, 84 - Utrera - Sevilla
Descripcion: Sur

Almacen 4 - Avenida de Portugal, 39 - Cuidad Rodrigo - Salamanca
Descripcion: Oeste

Almacen 5 - Calle San Mateo, 6 - Oviedo - Asturias
Descripcion: Norte

Almacen 6 - Calle de las flores, 4 - Priego - Cuenca
Descripcion: Serrania

Almacen 7 - Calle Rodrigo Diaz, 15 - Villarcayo - Burgos
Descripcion: Las Merindades

Almacen 8 - Calle Moron, 21 - Pastrana - Guadalajara
Descripcion: Alcarria

Almacen 9 - Avenida Iniesta, 3 - Plasencia - Caceres
Descripcion: Extremadura

Almacen 10 - Calle Sol, 1 - Ibi - Alicante
Descripcion: Alm Alc
```

4.- Alta de nuevo paciente en almacén sin pacientes.

Descripción de la prueba

Se da de alta a un paciente en un almacén sin pacientes si se comprueba su correcto registro y correcta muestra por pantalla de la lista de pacientes del almacén antes y después del registro.

Entrada de datos

Ricardo Sánchez; 1000, 1000

The screenshot shows a terminal window with the following text:

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma... X

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen             <Pulsar M>
    Alta paciente almacen   <Pulsar A>
    Ubicar pacientes         <Pulsar U>
    Nuevo pedido              <Pulsar N>
    Lista diaria de pedidos  <Pulsar L>
    Programar rutas diarias del dron  <Pulsar P>
    Representar rutas diarias del dron  <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <IIMIAUINILPIRIS>?
U

Lista de pacientes y su ubicacion:
    Codigo almacen? <entre 1-10> 2
    El almacen seleccionado no tiene pacientes registrados

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes           <Pulsar U>
    Nuevo pedido               <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I|M|A|U|N|L|P|R|S>?
A

Alta nuevo paciente:
    Codigo almacen? <entre 1-10> 2
    Alta nuevo paciente. Referencia paciente: 1
        Identificador <entre 1 y 20 carateres>: Ricardo Sanchez
        Distancia <hasta 10000 metros a plena carga>: 1000
        Angulo <entre 0 y 2000 pi / 1000 radianes>: 1000
    Los datos son correctos? <S/N>? S
    Otro paciente mismo almacen? <S/N>? N
    Otro Paciente <S/N>?N

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes           <Pulsar U>
    Nuevo pedido               <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I|M|A|U|N|L|P|R|S>?
U

Lista de pacientes y su ubicacion:
    Codigo almacen? <entre 1-10> 2
    Lista de pacientes y su ubicacion:
        Ref. Identificador          Distancia      Angulo
        1    Ricardo Sanchez         1000          1000

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

5.- Alta de nuevos pacientes en distintos almacenes

Descripción de la prueba

En una misma ejecución de la opción se da de alta a dos pacientes en distintos almacenes y se comprueba el correcto registro de los datos en cada almacén.

Entrada de datos

Almacén 2: Ataulfo Argenta; 567, 890

Almacén 3: Óscar Sevilla; 756, 98

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Lista de pacientes y su ubicacion:
Código almacén? <entre 1-10> 2
Lista de pacientes y su ubicacion:
Ref. Identificador           Distancia     Ángulo
  1   Ricardo Sanchez        1000         1000

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
  Iniciar gestion          <Pulsar I>
  Alta almacen              <Pulsar M>
  Alta paciente almacen    <Pulsar A>
  Ubicar pacientes          <Pulsar U>
  Nuevo pedido              <Pulsar N>
  Lista diaria de pedidos   <Pulsar L>
  Programar rutas diarias del dron <Pulsar P>
  Representar rutas diarias del dron <Pulsar R>
  Salir                      <Pulsar S>
Teclear una opcion valida <I|M|A|U|N|L|P|R|S>?
U

Lista de pacientes y su ubicacion:
Código almacén? <entre 1-10> 3
El almacén seleccionado no tiene pacientes registrados

Desea volver al menu GESTION DE FarmaDrones? <S/N>
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma... X

Alta nuevo paciente:
    Codigo almacen? <entre 1-10> 2
Alta nuevo paciente. Referencia paciente: 2
    Identificador <entre 1 y 20 caracteres>: Ataulfo Argenta
    Distancia <hasta 10000 metros a plena carga>: 567
    Angulo <entre 0 y 2000 pi / 1000 radianes>: 890
Los datos son correctos? <S/N>? S
Otro paciente mismo almacen? <S/N>? N
Otro Paciente <S/N>?S
Codigo almacen? <entre 1-10> 3
Alta nuevo paciente. Referencia paciente: 1
    Identificador <entre 1 y 20 caracteres>: Oscar Sevilla
    Distancia <hasta 10000 metros a plena carga>: 756
    Angulo <entre 0 y 2000 pi / 1000 radianes>: 98
Los datos son correctos? <S/N>? S
Otro paciente mismo almacen? <S/N>? N
Otro Paciente <S/N>?N

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes          <Pulsar U>
    Nuevo pedido               <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I!M!A!U!N!L!P!R!S>?
U

Lista de pacientes y su ubicacion:
    Codigo almacen? <entre 1-10> 2
Lista de pacientes y su ubicacion:
    Ref.  Identificador        Distancia     Angulo
    1    Ricardo Sanchez       1000         1000
    2    Ataulfo Argenta       567          890
```

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
    Angulo <entre 0 y 2000 pi / 1000 radianes>: 98
Los datos son correctos? <S/N>? S
Otro paciente mismo almacen? <S/N>? N
Otro Paciente <S/N>?N

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes          <Pulsar U>
    Nuevo pedido              <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I!M!A!U!N!L!P!R!S>?
U

Lista de pacientes y su ubicacion:
    Codigo almacen? <entre 1-10> 2
Lista de pacientes y su ubicacion:
    Ref. Identificador        Distancia      Angulo
    1    Ricardo Sanchez       1000           1000
    2    Ataulfo Argenta       567            890

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
    Iniciar gestion          <Pulsar I>
    Alta almacen              <Pulsar M>
    Alta paciente almacen    <Pulsar A>
    Ubicar pacientes          <Pulsar U>
    Nuevo pedido              <Pulsar N>
    Lista diaria de pedidos   <Pulsar L>
    Programar rutas diarias del dron <Pulsar P>
    Representar rutas diarias del dron <Pulsar R>
    Salir                      <Pulsar S>
Teclear una opcion valida <I!M!A!U!N!L!P!R!S>?
U

Lista de pacientes y su ubicacion:
    Codigo almacen? <entre 1-10> 3
Lista de pacientes y su ubicacion:
    Ref. Identificador        Distancia      Angulo
    1    Oscar Sevilla         756            98

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

6.- Registro de pedido en un paciente no registrado en el almacén.

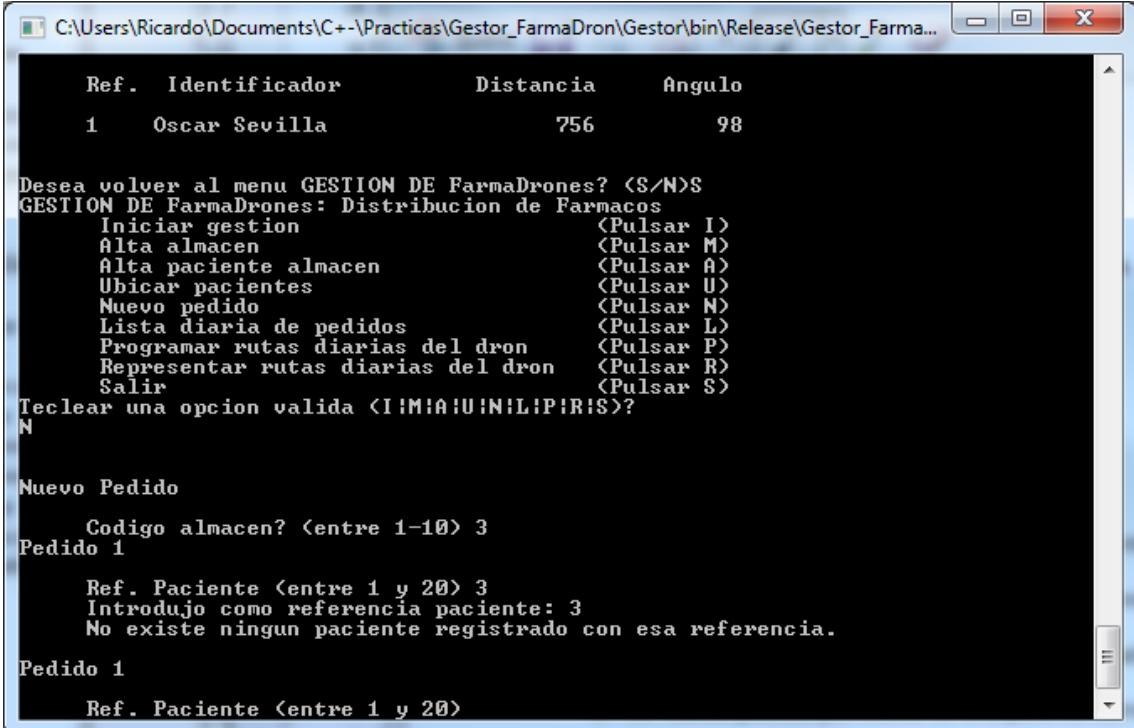
Descripción de la prueba

Se intenta registrar un pedido en un paciente que no está registrado en el almacén y se comprueba que la aplicación no lo permite.

Entrada de datos

En esta prueba no se introducen datos.

Capturas de salida



The screenshot shows a terminal window titled 'C:\Users\Ricardo\Documents\C+\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...'. The application displays a table of stored items:

Ref.	Identificador	Distancia	Angulo
1	Oscar Sevilla	756	98

Then, it asks if the user wants to return to the 'GESTION DE FarmaDrones' menu. The user types 'N' (No). The application then prompts for a new order:

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
GESTION DE FarmaDrones: Distribucion de Farmacos
Iniciar gestion <Pulsar I>
Alta almacen <Pulsar M>
Alta paciente almacen <Pulsar A>
Ubicar pacientes <Pulsar U>
Nuevo pedido <Pulsar N>
Lista diaria de pedidos <Pulsar L>
Programar rutas diarias del dron <Pulsar P>
Representar rutas diarias del dron <Pulsar R>
Salir <Pulsar S>
Teclear una opcion valida <I|M|A|U|N|L|P|R|S>?
N

Nuevo Pedido
Codigo almacen? <entre 1-10> 3
Pedido 1
Ref. Paciente <entre 1 y 20> 3
Introdujo como referencia paciente: 3
No existe ningun paciente registrado con esa referencia.

Pedido 1
Ref. Paciente <entre 1 y 20>

7.- Registro de pedido puntual de un único medicamento.

Descripción de la prueba

Se registra el pedido puntual de un único medicamento y se comprueba la correcta muestra por pantalla del registro.

Entrada de datos

Código de almacén: 1.

Referencia de paciente: 1.

Número de envíos: 1.

Día del envío: 1

Mes del envío: 2

Año del envío: 2021.

Nombre del fármaco: Aspirina.

Peso del fármaco: 123.

Unidades de fármaco: 2.

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
ListadoGestion_FarmaDron

Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripcion: Centro

Lista de pacientes y su ubicacion:
Ref. Identificador           Distancia     Angulo
1   Maria Carrera            5432         1990

Pedidos:
Cliente    Fecha        Farmaco      Peso      Unidades
1          21/1/2021    Bicarbonato  2750       1

Almacen 2 - Calle Acacias, 27 - Requena - Valencia
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Lista diaria de pedidos:
    Codigo almacen? <entre 1-10> 1
Pedido ALMACEN - CENTRO

    Dia? 1
    Mes? 2
    Ano? 2021

    Pedido: 1
Ubicacion destino: Distancia: 5432 y Angulo: 1990
    2 Unidades      Aspirina      Peso: 123 gramos
                                Peso Total del envio: 246 gramos

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

8.- Registro de pedido puntual de un único medicamento una fecha que ya tiene medicamentos registrados.

Descripción de la prueba

Se registra el pedido puntual de un único medicamento una fecha que ya tiene medicamentos registrados y se comprueba que el nuevo registro no se hace sobre el registro anterior y la correcta muestra por pantalla del registro.

Entrada de datos

Código de almacén: 1.

Referencia de paciente: 1.

Número de envíos: 1.

Día del envío: 21

Mes del envío: 1

Año del envío: 2021.

Nombre del fármaco: Aspirina.

Peso del fármaco: 123.

Unidades de fármaco: 2.

```
L

Lista diaria de pedidos:
 Codigo almacen? <entre 1-10> 1
Pedido ALMACEN - CENTRO

  Dia? 21
  Mes? 1
  Ano? 2021

  Pedido: 1
  Ubicacion destino: Distancia: 5432 y Angulo: 1990
  1 Unidades   Bicarbonato   Peso: 2750 gramos
  Peso Total del envio: 2750 gramos

Desea volver al menu GESTION DE FarmaDrones? <S/N>
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Pedido ALMACEN - CENTRO

Dia? 21
Mes? 1
Año? 2021

Pedido: 1
Ubicacion destino: Distancia: 5432 y Angulo: 1990
1 Unidades      Bicarbonato      Peso: 2750 gramos
                           Peso Total del envio: 2750 gramos

Pedido: 2
Ubicacion destino: Distancia: 5432 y Angulo: 1990
2 Unidades      Aspirina       Peso: 123 gramos
                           Peso Total del envio: 246 gramos

Desea volver al menu GESTION DE FarmaDrones? <S/N>
```

9.- Registro de un pedido periódico

Descripción de la prueba

Se registra el pedido periódico y se comprueba la correcta muestra por pantalla del registro.

Entrada de datos

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Nuevo Pedido
Código almacén? <entre 1-10> 1
Pedido 3
Ref. Paciente <entre 1 y 20> 1
Número de envíos? 2
Número de días entre cada envío <Entre 1 y 15 días>? 1
Día del primer envío? 20
Mes del primer envío? 1
Año del primer envío? 2021
Nombre del fármaco <Entre 1 y 20 caracteres>? Ibuprofeno
Peso fármaco <Menor de 3000 gramos>? 321
Unidades de fármaco? 3
Otro fármaco <S/N>? S
Nombre del fármaco <Entre 1 y 20 caracteres>? Omeprazol
Peso fármaco <Menor de 3000 gramos>? 678
Unidades de fármaco? 2
Otro fármaco <S/N>? N
```

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Pedido ALMACEN - CENTRO

Día? 21
Mes? 1
Año? 2021

Pedido: 1
Ubicación destino: Distancia: 5432 y Ángulo: 1990
1 Unidades      Bicarbonato      Peso: 2750 gramos
                                         Peso Total del envío: 2750 gramos

Pedido: 2
Ubicación destino: Distancia: 5432 y Ángulo: 1990
2 Unidades      Aspirina       Peso: 123 gramos
                                         Peso Total del envío: 246 gramos

Desea volver al menú GESTIÓN DE FarmaDrones? <S/N>
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Pedido ALMACEN - CENTRO

Dia? 20
Mes? 1
Ano? 2021

Pedido: 1
Ubicacion destino: Distancia: 5432 y Angulo: 1990
3 Unidades Ibuprofeno Peso: 321 gramos
2 Unidades Omeprazol Peso: 678 gramos
Peso Total del envio: 2319 gramos

Desea volver al menu GESTION DE FarmaDrones? <S/N>S
```

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Codigo almacen? <entre 1-10> 1

Pedido ALMACEN - CENTRO

Dia? 21
Mes? 1
Ano? 2021

Pedido: 1
Ubicacion destino: Distancia: 5432 y Angulo: 1990
1 Unidades Bicarbonato Peso: 2750 gramos
Peso Total del envio: 2750 gramos

Pedido: 2
Ubicacion destino: Distancia: 5432 y Angulo: 1990
2 Unidades Aspirina Peso: 123 gramos
Peso Total del envio: 246 gramos

Pedido: 3
Ubicacion destino: Distancia: 5432 y Angulo: 1990
3 Unidades Ibuprofeno Peso: 321 gramos
2 Unidades Omeprazol Peso: 678 gramos
Peso Total del envio: 2319 gramos
```

10.- Programar rutas del dron con un solo pedido.

Descripción de la prueba

Se registra un solo pedido en una fecha y se chequea la correcta programación de la ruta.

Aprovechando que se trata de un viaje de ida y vuelta se chequea el desfase de 180º en el ángulo de la ruta del dron. Los 180 grados en el contexto de la aplicación se representan con diferencias de 1000.

Entrada de datos

```
C:\Users\Ricardo\Documents\C++\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
ListadoGestion FarmaDron

Almacen 1 - Calle Manzano, 5 - Majadahonda - Madrid
Descripcion: Centro

Lista de pacientes y su ubicacion:
Ref. Identificador Distancia Angulo
1 Maria Carrera 5432 1990

Pedidos:
Cliente Fecha Farmaco Peso Unidades
1 21/1/2021 Bicarbonato 2750 1

Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripcion: Este
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C++\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Codigo almacen? <entre 1-10> 1
Dia? 21
Mes? 1
Ano? 2021

Ruta 1
Origen a Cliente1 -- Distancia recorrida: 5432 Angulo: 1990 Peso: 2750
Cliente1 a Origen -- Distancia recorrida: 5432 Angulo: 990 Peso: 0
Distancia total de la ruta: 10864

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

11.- Programar rutas del dron. Exceso de peso en la ruta

Descripción de la prueba

Se registran pedidos para el mismo día que se encuentren en el radio de acción del dron para una ruta, pero que el exceso de peso le haga volver al almacén y reiniciar la ruta.

Entrada de datos

Lista de pacientes y su ubicacion:				
Ref.	Identificador	Distancia	Angulo	
1	Maria Carrera	5432	1990	
2	Loreto Arenillas	7430	1255	
3	Jesus Aguado	2965	54	
4	Luis Miguel Brasero	3754	1934	
5	Carolina Moreno	23	578	

Pedidos:					
Cliente	Fecha	Farmaco	Peso	Unidades	
1	21/1/2021	Bicarbonato	2750	1	
1	21/1/2021	Ibuprofeno	250	2	
1	1/1/2021	Paracetamol	75	2	
2	1/1/2021	Antihistaminico	100	1	
2	30/1/2021	Omeprazol	350	2	
2	20/2/2021	Ibuprofeno	50	3	
3	1/1/2021	Alcohol	150	3	
3	1/1/2021	Aspirina	50	2	
3	10/1/2021	Omeprazol	750	1	

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
El codigo 21 no se corresponde con ningun almacen registrado
Codigo almacen? <entre 1-10> 1

Dia? 21
Mes? 1
Año? 2021

Ruta 1
Origen a Cliente1 -- Distancia recorrida: 5432 Angulo: 1990 Peso: 500
Cliente1 a Origen -- Distancia recorrida: 5432 Angulo: 990 Peso: 0
Distancia total de la ruta: 10864

Ruta 2
Origen a Cliente1 -- Distancia recorrida: 5432 Angulo: 1990 Peso: 2750
Cliente1 a Origen -- Distancia recorrida: 5432 Angulo: 990 Peso: 0
Distancia total de la ruta: 10864

Desea volver al menu GESTION DE FarmaDrones? <S/N>_
```

12.- Programar rutas del dron. Vuelta al origen por falta de autonomía y pedido hecho sin necesidad de volver al almacén.

Descripción de la prueba

Se sitúan los pedidos en clientes que obliguen al dron volver al almacén por falta de autonomía y pedidos en clientes que le permita hacerlos sin volver al almacén para comprobar el correcto funcionamiento del algoritmo.

Entrada de datos

Lista de pacientes y su ubicacion:				
Ref.	Identificador	Distancia	Angulo	
1	Maria Carrera	5432	1990	
2	Loreto Arenillas	7430	1255	
3	Jesus Aguado	2965	54	
4	Luis Miguel Brasero	3754	1934	
5	Carolina Moreno	23	578	

Pedidos:				
Cliente	Fecha	Farmaco	Peso	Unidades
1	21/1/2021	Bicarbonato	2750	1
1	21/1/2021	Ibuprofeno	250	2
1	1/1/2021	Paracetamol	75	2
2	1/1/2021	Antihistaminico	100	1
2	30/1/2021	Omeprazol	350	2
2	20/2/2021	Ibuprofeno	50	3
3	1/1/2021	Alcohol	150	3
3	1/1/2021	Aspirina	50	2
3	10/1/2021	Omeprazol	750	1

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Programar rutas diarias del dron:
Codigo almacen? <entre 1-10> 1
Dia? 1
Mes? 1
Año? 2021

Ruta 1
Origen a Cliente2 -- Distancia recorrida: 7429 Angulo: 1255 Peso: 100
Cliente2 a Origen -- Distancia recorrida: 7429 Angulo: 255 Peso: 0
Distancia total de la ruta: 14859

Ruta 2
Origen a Cliente1 -- Distancia recorrida: 5432 Angulo: 1990 Peso: 700
Cliente 1 a Cliente3 -- Distancia recorrida: 2595 Angulo: 916 Peso: 550
Cliente3 a Origen -- Distancia recorrida: 2965 Angulo: 1054 Peso: 0
Distancia total de la ruta: 10992
```

13.- Programar rutas del dron. Fusión de pedidos en un único viaje.

Descripción de la prueba

En una misma fecha se colocan distintos pedidos para un mismo cliente de forma que el algoritmo que programa las rutas del dron deba fusionar los pedidos y tratarlos a efectos de programación de rutas como un único viaje.

Entrada de datos

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Almacen 2 - Calle Acacias, 27 - Requena - Valencia
Descripcion: Este

Lista de pacientes y su ubicacion:
Ref. Identificador Distancia Angulo
1 Jose Maria Barrio 9034 1452
2 Eva Bailen 11789 378
3 Marta Llorente 334 250
4 Sergio Arribas 7113 994
5 Isabel Royo 643 12

Pedidos:
Cliente Fecha Farmaco Peso Unidades
1 30/3/2021 Ibuprofeno 25 2
1 30/3/2021 Bicarbonato 250 1
1 30/3/2021 Omeprazol 850 1
2 30/3/2021 Aspirina 115 2
2 22/2/2021 Bicarbonato 450 1
2 11/2/2021 Almax 150 4
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Programar rutas diarias del dron:
Codigo almacen? <entre 1-10> 2
Dia? 30
Mes? 3
Año? 2021

Ruta 1
Origen a Cliente1 -- Distancia recorrida: 9034 Angulo: 1452 Peso: 1150
Cliente1 a Origen -- Distancia recorrida: 9034 Angulo: 452 Peso: 0
Distancia total de la ruta: 18068

Ruta 2
Origen a Cliente2 -- Distancia recorrida: 11789 Angulo: 378 Peso: 230
Cliente2 a Origen -- Distancia recorrida: 11789 Angulo: 1378 Peso: 0
Distancia total de la ruta: 23578
```

14.- Programar rutas del dron. Fusión de algunos pedidos en un único viaje.

Descripción de la prueba

En una misma fecha se colocan distintos pedidos para un mismo cliente de forma que el algoritmo que programa las rutas del dron deba fusionar los pedidos, pero no todos, y tratarlos a efectos de programación de rutas como un único viaje hasta donde se lo permitan sus limitaciones.

Entrada de datos

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Descripción: Este

Lista de pacientes y su ubicacion:
Ref. Identificador           Distancia     Ángulo
1   Jose Maria Barrio       9034         1452
2   Eva Bailen              11789        378
3   Marta Llorente          334          250
4   Sergio Arribas          7113         994
5   Isabel Royo              643          12

Pedidos:
Cliente  Fecha      Farmaco    Peso  Unidades
1        30/3/2021  Ibuprofeno  25    2
1        30/3/2021  Bicarbonato  250   1
1        30/3/2021  Omeprazol   850   1
2        30/3/2021  Aspirina    115   2
2        22/2/2021   Bicarbonato  450   1
2        11/2/2021   Almax       150   4
1        30/3/2021  Mercromina  1400  2
```

Capturas de salida

```
C:\Users\Ricardo\Documents\C+-\Practicas\Gestor_FarmaDron\Gestor\bin\Release\Gestor_Farma...
Dia? 30
Mes? 3
Año? 2021

Ruta 1
Origen a Cliente1 -- Distancia recorrida: 9034 Ángulo: 1452 Peso: 1150
Cliente1 a Origen -- Distancia recorrida: 9034 Ángulo: 452 Peso: 0
Distancia total de la ruta: 18068

Ruta 2
Origen a Cliente1 -- Distancia recorrida: 9034 Ángulo: 1452 Peso: 2800
Cliente1 a Origen -- Distancia recorrida: 9034 Ángulo: 452 Peso: 0
Distancia total de la ruta: 18068

Ruta 3
Origen a Cliente2 -- Distancia recorrida: 11789 Ángulo: 378 Peso: 230
Cliente2 a Origen -- Distancia recorrida: 11789 Ángulo: 1378 Peso: 0
Distancia total de la ruta: 23578
```

4.- Conclusiones.

Conclusiones y logros en el ámbito formativo.

Con independencia de cómo se evalúe la práctica, he aprendido a programar y he hecho una aplicación desde cero. El cinco de octubre de 2020, hace 4 meses, no podría haber escrito ninguna de esas dos frases sin faltar a la verdad.

Yendo a aspectos concretos, manejé conceptos por *pasar parámetros por referencia*, *programación modular*, *tipo abstracto de datos* o *variable dinámica* que me parecen tener una extraordinaria potencia. Y lo que es más importante, despertó mi interés por saber más sobre el tema.

Conclusiones y logros en el ámbito personal.

Nietzsche comienza *Así habló Zaratustra* hablando de las metamorfosis del hombre; de camello en león, de león en niño.

El camello carga con los conceptos, el león ataca los conceptos y el niño juega con los conceptos. Pienso que es la transformación que viví con las prácticas de la asignatura; ahora me siento ante el ordenador para *jugar con los conceptos* como un niño practica con un juego de construcciones.

Muchas asignaturas las estudié porque era lo que tenía que hacer, con esta asignatura, construyendo algo y viendo que funcionaba, realmente disfruté. Para mí enfrentarme a los problemas de la práctica e irlos resolviendo fue una forma de entretenimiento y satisfacción personal.

Conclusiones y logros en el ámbito profesional.

Saber programar ya se puede considerar un logro en el ámbito profesional, pero no creo que sea ése el mayor logro que haya tenido con esta asignatura.

Profesionalmente es muy importante dominar las herramientas que se manejan y gracias a estas prácticas comprendo mucho mejor el lenguaje y los conceptos con los que están hechas las herramientas de cualquier profesional.

Hacer estas prácticas sería interesante para un médico en su ámbito profesional porque los conceptos que se manejan, y la puesta que abre a seguir profundizando en ellos, están en la base herramientas profesionales que debe manejar.