Fundamentos de Programación Sesión 10

Actividades a realizar en casa

Actividad: Resolución de problemas.

Resolved los siguientes problemas de la relación 4. Recordad que, **ANTES** del inicio de esta sesión en el aula de ordenadores, hay que subir las soluciones a PRADO. Se debe subir un fichero llamado sesion10.zip que incluya todos los ficheros cpp (y solamente estos).

15 (Carácter en una cadena)

16 (Moda)

17 (Pares-impares)

18 (Polígono)

19 (ADN)

Inspección de objetos

En esta sesión de prácticas vamos a trabajar sobre la manera en la que Dev C++ presenta la información sobre las clases definidas en el programa sobre el que se está trabajando y las facilidades que ofrece para monitorizar la ejecución de programas que incluyen clases. Usaremos como ejemplo una versión simple de la clase SegmentoDirigido que se puede encontrar en los apuntes de teoría. En primer lugar, crearemos la carpeta 10_SegmentoDirigido en U:\FP y copiaremos en ella el fichero fuente 10_SegmentoDirigido.cpp que se puede encontrar en la web de PRADO.

Emplearemos de nuevo el explorador de clases: seleccionar Ver | Ir al Explorador de Clases (o simplemente Ctrl+F3). En la Figura 1 puede observar que el explorador de clases muestra, para este programa, información acerca de los componentes de la clase SegmentoDirigido, definida en el fichero fuente abierto en el editor. También muestra las cabeceras de las funciones definidas en el fichero, cosa que ya conocemos del guión anterior (en este caso, únicamente la función main).

```
Proyecto Clases Depurar
                                                 III_SegmentoDirigido.cpp
SegmentoDirigido: dass
                                                       #include <iostream>
                                                 11
    --- 🍿 FinalAbscisa () : double
                                                 12
                                                       #include <cmath>
     - FinalOrdenada () : double
                                                 13
                                                       using namespace std;
     --- 🍿 Longitud () : double
                                                 14
    .... origenAbscisa (): double
                                                 15
                                                 16 ☐ class SegmentoDirigido{
    ... origenOrdenada (): double
                                                       private:
                                                 17
    SegmentoDirigido (double origen_abscisa, dou
                                                 18
                                                           double x_1, y_1, x_2, y_2;
    --- 🗽 SetCoordenadas (double origen_abscisa, double
                                                 19
    .... ..... Traslada (double unidades horiz, double unidade
                                                 20
                                                           SegmentoDirigido(double origen_abscisa,
    TrasladaHorizontal (double unidades) : void
                                                 21
                                                                             double origen_ordenada,
      TrasladaVertical (double unidades): void
                                                 22
                                                                             double final abscisa,
                                                                             double final_ordenada){
     ....priv x_1 : double
                                                 23
                                                 24
                                                              if (!SetCoordenadas(origen_abscisa, origen_ordenada,
     priv x 2 : double
                                                 25
                                                                                     final_abscisa, final_ordenada)){
     26
                                                                  x_1 = y_1 = x_2 = y_2 = 0
    ..... y_2 ; double
                                                 27
    main (): int
                                                 28
                                                 29
                                                 30
                                                           bool SetCoordenadas(double origen_abscisa,
                                                 31
                                                                                 double origen ordenada,
                                                 32
                                                                                 double final abscisa,
                                                 33
                                                                                 double final ordenada){
                                                 34
                                                              bool es_correcto;
```

Figura 1: Explorador de clases

En primer lugar observa que el explorador de clases organiza los elementos que muestra de manera jerárquica, por niveles. En este ejemplo encontramos dos elementos en el nivel principal (por orden de aparición en el explorador de clases):

- La clase SegmentoDirigido.
- La función main.

Se etiqueta al elemento Segmento Dirigido con la palabra class. Observa que se enumeran una serie de elementos dentro de esa clase que gráficamente se muestran asociados a Segmento Dirigido mediante líneas.

Dev C++ organiza la presentación de los elementos de la clase enumerando en primer lugar los métodos (por orden alfabético) y a continuación los datos (también por orden alfabético).

Haciendo click sobre el nombre de cualquier componente de la clase el editor muestra el código de la función seleccionada (su implementación o definición), o se accede a la declaración del dato seleccionado. Es una manera rápida de acceder al código de cualquier elemento de una clase.

Dev C++ usa un conjunto de iconos para mostrar los diferentes elementos del programa:



Los objetos, como cualquier otro dato, pueden ser monitorizados durante la depuración de un programa. Cuando se añade un objeto para su inspección se muestran sus campos y sus valores. Por ejemplo, en la Figura 2 se muestra el contenido del objeto un_segmento después de haber asignado los valores a sus cuatro campos.

```
[*] III_SegmentoDirigido.cpp
Proyecto Clases Depurar
un_segmento =
                               74 | };
     76
     y_1 = 5.5999999999999999
                               77
     x_2 = 4.5
                               78 ☐ int main(){
     y_2 = 2.2999999999999998
                                        SegmentoDirigido un_segmento(3.4, 5.6, 4.5, 2.3);
                               79
                               80
                                        cout << "\nLongitud del segmento = " << un_segmento.Longitud();</pre>
                               1
                               82
                               83
                                        cout << "\n\n";
                               84
                                        cout << system("pause");
```

Figura 2: Explorador de clases mostrando el contenido de un objeto SegmentoDirigido

Para llegar al estado mostrado en la Figura 2:

- 1. se estableció un punto de ruptura después de crear el objeto,
- 2. se añadió la variable un_segmento a la lista de datos monitorizados (botón derecho sobre un segmento y Añadir watch)
- 3. se inició la depuración del programa (Ejectuar | Depurar).

Una vez que aparece el objeto un_segmento entre los datos monitorizados puedes trabajar con él (y con cualquiera de los campos que lo forman) de la misma manera que con cualquier otro dato monitorizado.

Durante la depuración puedes avanzar paso a paso (**F8**), y entrar a ejecutar los métodos de la clase línea a línea, tal y como se vio en la sesión de prácticas con funciones. Por ejemplo, establece un punto de ruptura en la línea

```
un segmento.TrasladaHorizontal (10);
```

continúa la depuración hasta llegar a esa línea y pulsa **F8**. Cuando se está ejecutando el método TrasladaHorizontal, los cambios realizados por éste en los datos miembro no aparecen reflejados en la ventana de inspección del objeto un_segmento (para comprobarlo es posible que se tenga que mover el ratón sobre el código para que se refresque la ventana de inspección). Si estamos ejecutando un método y queremos ver cómo cambian los datos miembro, debemos añadir explícitamente una inspección para los datos miembros (ver Figura 3).

```
[*] III_SegmentoDirigido.cpp
Proyecto Clases Depurar
                                     62 <del>|</del> 63 <del>|</del>
  un_segmento = Not found in curi
                                               void TrasladaHorizontal(double unidades){
  x 1 = x 1 + unidades;

x 2 = x 2 + unidades;
                                     64
  y_1 = 5.5999999999999999
                                     $5
  x_2 = 4.5
                                     66
   y_2 = 2.29999999999999998
                                     67 🖨
                                               void TrasladaVertical(double unidades){
                                                  y_1 = y_1 + unidades;
y_2 = y_2 + unidades;
                                     69
                                     70
                                               void Traslada(double unidades_horiz, double unidades_vert){
                                     71 日
                                                  TrasladaHorizontal(unidades_horiz);
                                     72
                                                  TrasladaVertical(unidades_vert);
                                     73
                                     74
                                     75 [ };
                                     77 ☐ int main(){
                                               SegmentoDirigido un_segmento(3.4, 5.6, 4.5, 2.3);
                                     78
                                     79
                                     80
                                               cout << "\nLongitud del segmento = " << un_segmento.Longitud();</pre>
                                     81
                                     83
                                     84
                                               cout << "\n\n";
                                     85
                                               cout << system("pause");
                                     86
```

Figura 3: Explorador de clases mostrando los datos miembro durante la depuración

Al salir del método y volver a la función main comprobaremos que el objeto un_segmento se actualiza y se muestran los valores correctos.

El programa presenta un pequeño error lógico. Encontrar dicho error utilizando el depurador.

Actividad: Resolución de problemas.

Resolved los siguientes problemas de la relación 4B.

- 1 (Recta)
- 2 (DepositoSimulación)