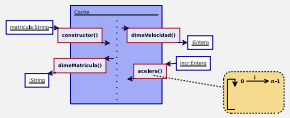
Grado Tecnologías Interactivas

Programación 1



Práctica 4

UNIVERSIDAD POLITECNICA DE VALENCIA

Escola Politècnica Superior de Gandia

DSIC

Departament de Sistemes Informàtics i Computació

Objetivos

- General: diseño de clases/objetos.
- General: diseño de funciones que reciban o devuelvan objetos.
- General: diseño de funciones que reciban o devuelvan listas de objetos.
- General: algoritmos sencillos sobre listas.
- C/C++: uso de los anteriores conceptos en este lenguaje.

¡ Atención !

▷ Se recuerda que las prácticas deben prepararse antes de acudir al aula informática, anotando en el enunciado las dudas que se tengan.

▷ Los diseños y algortimos que se piden en esta práctica deben escribirse en la libreta de apuntes para poder ser revisados.

▷ La realización de las prácticas es un trabajo individual y original. En caso de plagio se excluirá al alumno de la asignatura. Por tanto, es preferible presentar el trabajo realizado por uno mismo aunque éste tenga errores.

1 Introducción a la orientación a objeto

Las aplicaciones y programas informáticos reciben información de su entorno o "mundo real" que procesan para obtener nueva información.

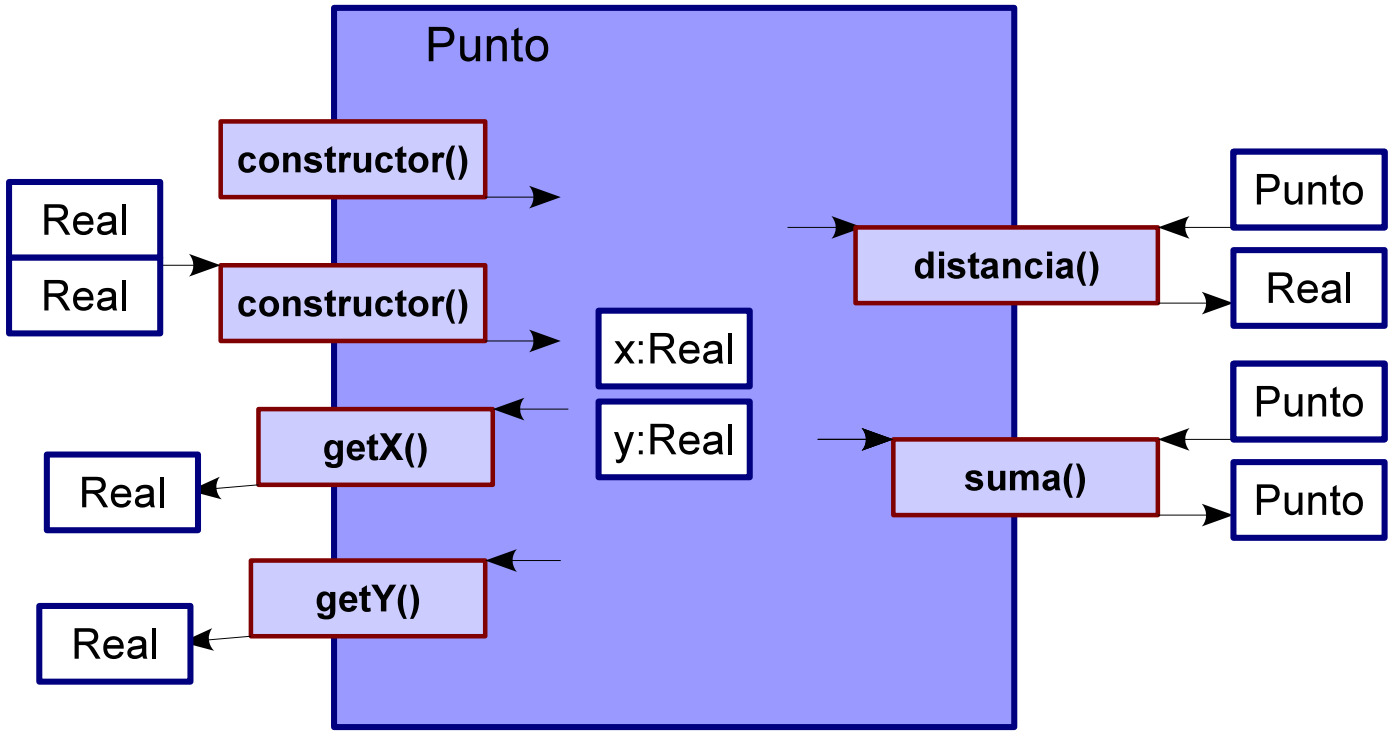
En este mundo real existen "objetos" que se caracterizan por guardar un estado (información) vinculado a varias operaciones para consultar o modificar dicho estado.

Así pues, resulta conveniente que los lenguajes de programación faciliten la representación de objetos del mundo real como una "unidad" y no como valores desagregados.

Por ejemplo, un punto en dos dimensiones es "una única cosa" que, por supuesto, tiene partes (la coordenada x y la coordenada y) pero que siempre "van juntas" puesto que son un "objeto punto".

Cuando necesitemos representar un tipo de objeto en un programa, primero diseñaremos la clase: cuál es la estructura común de los objetos de esta clase. Posteriormente, declararemos tantos objetos (= variables) como necesitemos de dicha clase.

Veamos el diseño de la clase Punto.



El objetivo de esta práctica es aprender a utilizar objetos de la clase Punto.

2 Ejercicios

Atención: los programas deben estar correctamente comentados. Antes de cada función debe haber una comentario que resuma el diseño de la misma. En el código del cuerpo de la función hay que intercalar los pasos del algoritmo.

Es obligatorio utilizar GIT.

1. Lee el fichero main.cpp. Prueba este programa. Aviso: forman parte también del programa los ficheros Punto.h y Punto.cpp que sirven para implementar el diseño de la clase Punto. Estos ficheros se estudiarán más adelante.
   La orden para compilar el programa (desde la línea de comandos) es

   ```
   g++  Punto.cpp main.cpp
   ```

2. Diseña, escribe su algoritmo e implementa una función llamada queCuadrante() (nombre del fichero con el main(): mainCuadrante.cpp) que reciba un punto y devuelva en qué cuadrante (1, 2, 3 o 4) del plano cartesiano se encuentra dicho punto.
   Escribe (en main()) una prueba automática de tu función para verificar que el punto $(-3, -8)$ está en el cuandrante 3.

3. Diseña, escribe su algoritmo e implementa una función llamada lejano() (nombre del fichero con el main(): mainLejano.cpp) que reciba dos puntos y devuelva el más lejano al origen.
   Escribe una prueba automática de tu función para verificar que el si la función recibe los puntos $(2, 3)$ y $(-2, -5)$ devuelve el punto $(-2, -5)$.

4. Diseña, escribe su algoritmo e implementa una función llamada `cercano()` (nombre del fichero con el `main()`: `mainCercano.cpp`) que reciba una lista de puntos y devuelva el más cercano al origen de todos ellos. Piensa y escribe una prueba automática para esta función.

5. Diseña, escribe su algoritmo e implementa una función llamada camino() (nombre del fichero con el main(): mainCamino.cpp) que reciba una lista de puntos y devuelva la longitud del camino que forman.
   Piensa y escribe una prueba automática para esta función.

10 octubre 2018