Programación y Estructuras de Datos

SEMINARIO C++

Guía rápida

SEMINARIO - SESIÓN 3

"C++ paso a paso"
Capítulo 4 : "Funciones, clases amigas y reserva de memoria"

- **4.2 Declaración de amistad**

  - Fíjate en los dos formatos de uso de "**friend**" : como función amiga y como clase amiga. (pág. 56)

  - Teclea y compila el **ejemplo 4.1** (añadiendo a **tcoordenada.h**) para ver cómo se integran una función **friend** ("Distancia") y una clase **friend (**"TLinea"**)** en nuestra clase de pruebas.

  - Fíjate sobre todo en 2 de los puntos de la definición de **friend** (pág. 57) :

    - (punto segundo) si la clase "B" declara amiga a la clase "A", entonces todos los métodos de "A" podrán acceder a la parte PRIVATE de "B" (ojo! NO al revés).
    - (punto sexto) distinguir bien quién "otorga" la amistad de quién la "recibe" : la otorga quien la declara explícitamente.

  - Teclea y compila el **ejemplo 4.3** (añadiendo a **tcoordenada.cc**) para ver cómo se integran la función **friend** ("Distancia") y la clase **friend (**"TLinea"**)** en nuestra clase de pruebas.

  - Teclea y compila el **ejemplo 4.4** (llamándolo **tlinea.h**), el **ejemplo 4.5** (llamándolo **tlinea.cc**) y el **ejemplo 4.6** (llamándolo **main.cc**). Observa 2 cosas :

    - Para poder compilar correctamente , deberás retocar el fichero MAKE que ya tenías, añadiendo "**tlinea.o**", "**tlinea.h**" y "**tlinea.cc**" donde proceda.
    - Comprende por qué la compilación da error : el compilador recibe varias veces la instrucción **#include "tcoordenada.h".**

- **4.3 Guardas de inclusión**

  - Fíjate en qué sitios del código compilado se confirma la observación de que el compilador recibe varias veces la instrucción **#include "tcoordenada.h"** (pág. 62).

  - Fíjate en las principales directivas para el PREPROCESADOR : #ifndef , #define, #endif (pág. 63).

  - Teclea y compila el **ejemplo 4.7** (completando **tcoordenada.h**) y el **ejemplo 4.8** (completando **tlinea.h**), añadiendo las directivas para el PREPROCESADOR.

- **4.4 Administración de memoria dinámica**

  - Fíjate en cómo se debe chequear la corrección de cada reserva de memoria (pág. 65 , *prtInt, *prtCoor), para evitar el error "*Segmentation fault*" .

  - Fíjate en cómo se debe chequear la corrección de cada liberación de memoria (pág. 65-66), para evitar el error "*Segmentation fault*" .

- **4.5 Administración de memoria dinámica y arrays de objetos**

  - Fíjate en el **ejemplo 4.10** (líneas 10 y 11)**,** y en cómo definir de 2 formas un ARRAY (pág.67).

  - Teclea y compila el **ejemplo 4.9** (añadiendo los 2 COUT) y el **ejemplo 4.10**, comprobando las salidas de los COUT (**salida ejemplo 4.10**) :

    - Las líneas 1 a 3 : corresponden al **array[3] .**
    - Las líneas 5 a 7 : corresponden al ***ptr.**
    - Las líneas 9 a 11 : corresponden al **array[3] .**

  - Teclea y compila el **ejemplo 4.12** (añadiendo líneas 18 a 22), comprobando las salidas de los COUT (**salida ejemplo 4.12,** líneas 9 a 11).

  - Fíjate en la importancia de usar la expresión "**delete []**" (pág.69).

- Por último, fíjate en la importancia de saber cuándo usar la expresión "**delete []**" y "**delete**" , en los 4 casos planteados (pág.70).

- **4.6 Compilación condicional**
  - Leer el apartado.

- **4.7 Directivas #WARNING y #ERROR**
  - Leer el apartado.

- **4.9 Ejercicios de PROGRAMACIÓN / 4.11 Respuesta a ejercicios de PROGRAMACIÓN**
  - El **ejemplo 4.9.2** propone la clase **TCalendario** .
  - La **solución 4.11.2** resuelve inicialmente lo que se pide de la clase **TCalendario .**