Universidad de Jaén
Departamento de Informática

Programación Orientada a Objetos

http://tiny.cc/poouja

Curso 2022/2023

José Ramón Balsas Almagro
Víctor Manuel Rivas Santos
Ángel Luis García Fernández
Juan Ruiz de Miras

Práctica 0. Herramientas de programación

Objetivos

- Saber instalar el entorno de programación y herramientas auxiliares.
- Saber utilizar el depurador para ejecutar instrucción a instrucción un programa y evaluar el valor de sus variables.
- Repasar el manejo de punteros y paso de parámetros.
- Saber generar la documentación externa de un proyecto.

Contenidos

Ejercicios

1.- (preparación de la práctica) Aunque en las aulas de informática ya se encuentran instalados los entornos y herramientas de desarrollo, es conveniente que **previamente** a la sesión de prácticas realices su instalación en tu ordenador personal y compruebes que

funcionan correctamente. Las dudas o problemas que te surjan durante el proceso puedes consultarlas con tu profesor de prácticas en la propia sesión. En el siguiente enlace puedes consultar el software necesario y algunas indicaciones sobre cómo instalarlo en diferentes sistemas operativos: http://tiny.cc/poouja_software. Si fuera necesario, también puedes utilizar como referencia los diferentes tutoriales existentes en los sitios webs de cada herramienta o en otros sitios de Internet.

2.- (preparación de la práctica) Repasa el uso del tipo puntero y referencial en C++. Estudia las diferentes formas de realizar un paso de parámetros por referencia a funciones utilizando tanto referencias como punteros de diferentes tipos de datos: tipos simples, estructuras de datos y vectores. Consulta los apuntes de *Programación en C++* de la asignatura de *Fundamentos de Programación* (apartado 7, punteros).

3.- Se desea desarrollar una aplicación para la gestión de los vehículos de un concesionario. Para cada vehículo se almacenará la siguiente información:

- *Marca*: cadena de caracteres (longitud máxima 20 y mínima 3).
- *Modelo*: cadena de caracteres (longitud máxima 30 y mínima 1).
- *Matrícula*: cadena de caracteres (longitud 6).
- *Año de fabricación*: entero.
- *Precio*: float (>0).

Define **un struct** llamado *Vehiculo* para representar esta información.

4.- Implementa **en C** y especifica usando sintaxis de Doxygen las siguientes funciones de apoyo para la aplicación:

- *leePorTeclado(Vehiculo)*: Pide al usuario que introduzca la información de un vehículo desde teclado, modificando con los nuevos datos los atributos del vehículo que se recibe como parámetro. Utiliza para esta función paso de parámetros por referencia.
- Realiza una versión sobrecargada de la anterior función que realice un paso de parámetros mediante un puntero.
- *muestraEnPantalla(Vehiculo)*: Muestra por pantalla la información almacenada en un *Vehiculo*. Realiza dos versiones sobrecargadas: por referencia y mediante un puntero.

5.- Apoyándonos en las funciones del ejercicio anterior, escribe una función *main* que haga lo siguiente:

- Definir un vector con un tamaño fijo suficientemente grande (es recomendable que definas una constante MAX_VEHICULOS) y solicite al usuario por teclado el número de vehículos a introducir (como máximo el tamaño del vector).

- Utilizar las funciones del ejercicio 4 para rellenar el vector y, posteriormente, mostrar su contenido por pantalla.
- Por último, buscar y mostrar por pantalla los datos del vehículo con mayor precio.

6.- Genera la documentación del proyecto en formato html utilizando la herramienta *doxywizard*. Prueba a marcar diferentes opciones de generación de la documentación y comprueba los resultados obtenidos: *Extraction mode all entities, include cross-reference source code, Output Language: Spanish*, etc.

Explicación de Conceptos

Entornos de programación en C/C++

Para el desarrollo de las prácticas de la asignatura utilizaremos el lenguaje de programación C++. Existen multitud de compiladores de C++ disponibles para las diferentes plataformas habituales: GNU/Linux, Mac y Windows, tanto comerciales como gratuitos. Puesto que el lenguaje que utilizaremos será C++ estándar, cualquier compilador de C++ servirá para realizar los ejercicios propuestos a lo largo de las prácticas.

En el aula de prácticas estarán disponibles las herramientas de desarrollo en C++ del proyecto de software libre GCC. Para el sistema operativo Windows, estas herramientas se han instalado desde el sistema *MinGW*. En GNU/Linux también están disponibles las herramientas de desarrollo en C++ de GCC, que pueden ser instaladas utilizando el gestor de paquetes de la propia distribución.

Para la plataforma **GNU/Linux**, se deberá instalar el entorno de compilación de C/C++ instalando los paquetes correspondientes dependiendo de la distribución que se esté utilizando. Por ejemplo, en los sistemas basados en Ubuntu se instalará el metapaquete ***build-essential*** mediante el comando:

```
sudo apt-get install build-essential
```

En **MacOS** (https://developer.apple.com/xcode/) se deberá instalar el entorno de desarrollo Xcode que suele venir en el segundo CD de instalación del sistema operativo, o bien se puede descargar desde la propia AppStore.

Puedes consultar más información sobre las herramientas que se utilizarán en prácticas en el documento de descripción de software empleado en la asignatura. A través de ese documento también dispones de un enlace a un videotutorial que presenta toda la instalación de las

herramientas en el entorno Windows.

Entornos de desarrollo integrados

Los entornos de desarrollo integrados son aplicaciones que ayudan al programador durante el proceso de desarrollo y depuración de un software. Disponen de utilidades para visualizar y editar diferentes aspectos del proyecto que se está desarrollando, además de interaccionar de forma transparente con herramientas externas que intervienen en el proceso de implementación (compiladores, depuradores, etc.).

Para la realización de las prácticas se ha elegido el entorno de desarrollo integrado *CLion* (https://www.jetbrains.com/es-es/clion), que es un IDE (*Integrated Development Environment*) al que puedes acceder con una licencia de uso educativo, y que puede ejecutarse en los sistemas operativos de escritorio más extendidos: GNU/Linux, MacOS y Windows. Una vez instalado, se pueden incorporar plugins a través de los menús de la propia aplicación (*File->Settings->Plugins*).

CLion será el entorno de referencia que utilizaremos a lo largo de la asignatura. No obstante, existen otros entornos de desarrollo integrados tanto gratuitos como de pago que son igualmente válidos para el desarrollo en C++, tales como *NetBeans (libre, multiplataforma), Eclipse* (libre, multiplataforma), *Visual Studio* (gratis, Windows), *Qt Creator* (libre, multiplataforma) o *Xcode* (gratis, MacOS) entre otros.

Una vez instalado el entorno de desarrollo en Windows, cuando se vaya a crear un nuevo proyecto en C++, habrá que seleccionar en el correspondiente IDE qué entorno de compilación/ejecución habrá que utilizar para compilar, ejecutar y depurar el proyecto. Puesto que en este proceso pueden intervenir gran cantidad de herramientas (gestor de dependencias, precompilador, compilador, ensamblador, enlazador, depurador, etc.), se suele utilizar el nombre *Toolchain* dentro del IDE para referirse al conjunto de todas las aplicaciones que forman parte de alguno de los entornos de desarrollo existentes. Por lo tanto, a la hora de compilar, ejecutar y depurar un proyecto desde CLion habrá que indicarle que usaremos el *Toolchain MinGW*. Normalmente los IDEs detectan automáticamente la presencia de estas herramientas, pero si no fuera así, existen opciones en cada uno de ellos para agregar estos *Toolchains* y que puedan ser utilizados por el propio entorno[1].

[1] Se pueden añadir de forma manual otros entornos de compilación desde el apartado File > Settings > Build, Execution, Deployment > Toolchains. Puede consultarse más información sobre el proceso en el siguiente vídeo

¡¡¡ATENCIÓN!!!
Dispones de un videotutorial que presenta la compilación y depuración de código con el IDE CLion en este enlace. (Utiliza tu cuenta @red.ujaen.es para acceder al vídeo)

Por último, es importante hacer un buen uso de las facilidades que proporcionan los entornos de programación para ser más productivos durante el proceso de desarrollo: atajos de teclado, opciones de formateo automático del código, autocompletar código, etc. Para conocer más detalles sobre estas características puedes consultar los tutoriales que traen los propios entornos. En estos tutoriales encontrarás información sobre cómo gestionar proyectos, editar código fuente, facilidades de ayuda durante la escritura de código, así como instrucciones sobre cómo realizar la compilación, ejecución y depuración. Puedes acceder a los tutoriales para trabajar con C/C++ en CLion a través de https://www.jetbrains.com/clion/learn. Dentro de estos tutoriales, puedes consultar los de Quick start guide, Open, reopen and close projects, Work with source code, y Debug.

Herramienta de ayuda para la documentación de código

Doxygen (http://www.doxygen.nl) es una herramienta de software libre que permite generar documentación del código en diferentes formatos (RTF, HTML, PostScript, XML, etc.) a partir de los comentarios introducidos por el programador en el propio código. Puede trabajar con varios lenguajes de programación, entre los cuales se encuentran C y C++. Existen versiones disponibles para los sistemas operativos de escritorio más habituales (GNU/Linux, MacOS y Windows). Es una utilidad en línea de comandos que debe ejecutarse en el directorio donde se encuentra el código a procesar, aunque normalmente se utiliza desde herramientas gráficas independientes (*doxywizard*).

Puedes descargar los instaladores de Doxygen para MacOS y Windows desde su página web, mientras que en las distribuciones de GNU/Linux, se puede instalar desde el gestor de paquetes. En los sistemas basados en Ubuntu, el paquete a instalar se llama *doxygen-gui*.

Una vez instalada la aplicación, es necesario colocar en la ruta de búsqueda de programas del sistema operativo (PATH) el directorio *bin* de la aplicación (si no se hubiera hecho automáticamente durante la instalación).

La aplicación *doxywizard*, incluida con la instalación de Doxygen, permite configurar el modo de funcionamiento del generador de documentación. Esta utilidad permite crear un fichero de configuración (cuyo nombre por defecto es *Doxyfile*) que podrá utilizarse para múltiples proyectos, personalizándolo adecuadamente en cada caso, ya sea con un editor de textos

simple (como el bloc de notas de Windows) o con la propia *Doxywizard*.

Una vez configuradas mediante *doxywizard* las diferentes opciones de funcionamiento, el formato de salida y seleccionados los ficheros y directorios que van a ser analizados, debemos guardar la configuración, normalmente en el mismo directorio donde se encuentran los ficheros fuente, y a continuación se lanza Doxygen. Esto se puede hacer desde el propio *doxywizard* o por consola, ejecutando el siguiente comando:

```
doxygen fichero_configuracion
```

Los documentos generados estarán en diferentes directorios según el formato seleccionado: RTF, HTML, etc. El formato más cómodo durante el desarrollo es el HTML, puesto que permite movernos cómodamente a través de diferentes listados a través de toda la documentación y estructura de la aplicación que se está desarrollando.

Sintaxis básica de los comentarios Doxygen

La utilidad ***Doxygen*** genera la documentación a partir de comentarios en el código que tienen el siguiente formato:

```
/** comentario */
/// comentario de una línea
```

Existen **etiquetas** adicionales para organizar y dar formato a estos comentarios según las necesidades. Por ejemplo, aquí puedes ver la documentación de una función:

```
/** @brief   Elimina personas con el mismo dni de un vector
    @author  Sheldon
    @date    31-12-2010
    @param   [in,out] v TPersona[]
    @param   tam int Longitud del vector v
    @pre     El vector debe estar inicializado
    @post    Elimina del vector la segunda y posteriores ocurrencias
             de personas con DNIs ya existentes
    @return  Devuelve el número de personas restantes que quedan en el
             vector
    @retval  -1 Si tam<0
    @note    No elimina la memoria sobrante del vector
    @see     personas.h
*/
int eliminaDuplicados(TPersona v[], int tam );
```

Además de servir para la documentación de funciones, también se pueden documentar los ficheros e incluso variables y estructuras de datos. Aquí tienes algunos ejemplos:

```
/**  @brief     Aplicación de prueba para algoritmos de ordenación
     @file      main.cpp
     @author    Sheldon Cooper
     @version   1.0
     @date      2-3-2011
*/
```

```
/**  @brief Información asociada a una persona*/
struct TPersona {
     string nombre;///<Nombre completo persona(sintaxis para campos)
     int edad;  ///<Edad de la persona
};
```

```
/**  @brief tamaño máximo del vector */
const int NUMDATOS=100;
```

```
/** Programa principal (sin etiqueta equivale a brief hasta la primera línea en blanco) */
int main ()  {
    //...
}
```