My Project

Generated by Doxygen 1.8.17

Chapter 1

Práctica 08. Funciones. GH Classroom. Doxygen.

1.1 Factor de ponderación: 7

1.1.0.1 Objetivos

Los objetivos de esta práctica son que el alumnado:

- Sea capaz de resolver problemas sencillos en C++ usando todos los conocimientos adquiridos hasta ahora, y en particular utilizando funciones
- Diseñe, desarrolle y utilice funciones en sus programas haciendo que sus programas sean modulares
- Comience a gestionar sus asignaciones de trabajos prácticos usando `GitHub Classroom`

1.1.0.2 Rúbrica de evaluacion de esta práctica

Se señalan a continuación los aspectos más relevantes (la lista no es exhaustiva) que se tendrán en cuenta a la hora de evaluar esta práctica. Se comprobará que el alumnado:

- Conoce los conceptos expuestos en el material de referencia de esta práctica.
- Ha realizado todos los ejercicios propuestos en este enunciado
- Es capaz de escribir programas simples en C++ que resuelvan problemas de complejidad similar a los que se proponen en este documento
- Ha automatizado la compilación de sus programas usando un fichero `Makefile` para cada uno de los programas que desarrolle
- Hace que sus programas se estructuren en torno a diferentes funciones (sean modulares)
- Conozca los fundamentos de la documentación de código
- Conozca la herramienta Doxygen, las etiquetas definidas en JSDoc y sepa cómo utilizarlas para documentar su código
- Acredita que todas las prácticas realizadas hasta la fecha se encuentran alojadas en repositorios privados de `GitHub`.
- Acredita que es capaz de subir programas a la plataforma `Jutge` para su evaluación

- Ha incluido un comentario prólogo en todos los ficheros (`*.cc`, `*.h`) de sus ejercicios
- Que todos los programas que desarrolla, antes de su ejecución imprimen en pantalla un mensaje indicando la finalidad del programa así como la información que precisará del usuario para su correcta ejecución.
- Hace que todos los programas que se presentan para su evaluación cumplan con los estándares definidos en la Guía de estilo de Google para C++
- Utiliza siempre identificadores significativos en su programa (para constantes, variables, etc.) y no utiliza nunca identificadores de una única letra, tal vez con la excepción de las variables que utilice para iterar en un bucle.
- Acredita que es capaz de editar ficheros remotos en su VM usando vi
- Ha realizado todos sus ejercicios en la máquina virtual Ubuntu de la asignatura.
- Demuestra que es capaz de ejecutar comandos Linux en su VM

1.1.0.3 GitHub Classroom

En el futuro se utilizará GitHub Classroom (una plataforma relacionada con GitHub) para gestionar las prácticas de *Informática Básica*. En esa plataforma, para la realización de cada práctica recibirá una invitación a una tarea que tendrá que aceptar. Una vez acepte la invitación tendrá que clonar con `git` un repositorio asociado a la tarea. Ese repositorio privado será el punto de partida y tendrá Ud. que añadir en él directorios con los programas que realice.

El enlace de invitación a la tarea que se le comunicará tiene una apariencia similar a https://classroom.github.com/a/uNbth9vD Si lo introduce en un navegador, se le solicitará que se autentifique en su cuenta de GitHub, y una vez autentificada/o le llevará a una pantalla como esta en la que se le solicitará que se una a la "∗classroom∗" IB-2023-2024. Para ello ha de seleccionar su nombre de la lista de identificadores (*Identifiers*) que figura en esa página. A continuación se le solicitará que "Acepte la tarea Practica-Nombre" Habrá una tarea asociada con cada una de las prácticas de la asignatura.

Cuando lo haya hecho aparecerá una pantalla como esta que indica que ha aceptado Ud. la tarea asignada y cuando refresque la pantalla le mostrará otra pantalla en la que figura el enlace al repositorio que ha sido creado para su trabajo en la práctica.

A través de ese enlace accederá Ud. en GitHub al repositorio privado que se ha creado para que desarrolle en él los programas correspondientes a la práctica en cuestión. El enlace que figura en ese repositorio (elija la opción SSH para el enlace) es el que ha de entregar Ud. en la tarea del aula virtual correspondiente a la práctica.

Recuerde que para cada práctica tiene que entregar 2 elementos:

- Este enlace a su repositorio. Ese enlace lo puede ya escribir en la tarea correspondiente del aula virtual: no es necesario que espere a la sesión de evaluación para subirlo.
- Un fichero `.tar.gz` conteniendo todos los programas que desarrolle tanto antes como durante la sesión de evaluación. Sí ha de esperar a la sesión de evaluación para subir el fichero .tar.gz conteniendo sus programas.

Ese mismo enlace es el que ha de utilizar para realizar una copia local (clone) del repositorio en su máquina virtual y comenzar a trabajar en los ejercicios de la práctica:

```
git clone https://github.com/IB-2023-2024/P08-functions <DirectorioLocal>
```

1.1.0.4 Doxygen

Doxygen es una herramienta de código abierto que permite generar documentación de referencia para proyectos de desarrollo software. Una ventaja de Doxygen es que la documentación está escrita en el propio código fuente de los programas, y por lo tanto es relativamente fácil de mantener actualizada. Doxygen puede hacer referencias cruzadas entre la documentación y el código, de modo que el lector de un documento puede referirse fácilmente al código fuente. La herramienta extrae la documentación de los comentarios presentes en los ficheros de código fuente y puede generar la salida en diferentes formatos entre los cuales están HTML, PDF, LaTeX o páginas man de Unix.

En *Informática Básica* no se propone un uso exhaustivo de Doxygen pero **sí se requiere que la documentación de los programas desarrollados se realice en el formato reconocido por Doxygen**, que se ha convertido en un estándar de facto.

Comience por instalar Doxygen en su máquina virtual de la asignatura:
```
$ sudo apt install doxygen
```

Instale también los siguientes paquetes:
```
$ sudo apt install texlive-latex-base
$ sudo apt install texlive-latex-recommended
$ sudo apt install texlive-latex-extra
```

Estos paquetes son necesarios para compilar ficheros en formato LaTeX. Más adelante en este documento se justifica la necesidad de los programas que suministran estos paquetes.

En el manual de Doxygen se indica cómo comenzar a trabajar con la herramienta. Si, ubicados en un directorio de trabajo, se invoca:
```
doxygen -g <config-file>
```

la herramienta creará un fichero de configuración. Si no se le pasa el nombre del fichero (*config-file*) como parámetro, creará un fichero con nombre Doxyfile preconfigurado para su uso. En el directorio de trabajo de esta práctica (src) se encuentra un fichero Doxyfile ya listo para usarse con proyectos de C++. Se ha incluído asimismo en ese directorio el código fuente de un programa para ilustrar con el mismo el uso de documentación con Doxygen. Si revisa el fichero Doxyfile (es un fichero de texto) verá un conjunto de opciones que el programa permite. Cada opción va precedida de una explicación de su finalidad y funcionamiento, de modo que puede probar a modificar algunas de ellas si lo desea. En esta página puede consultarse la finalidad y funcionamiento de cada una de las etiquetas (tags) que se usan en el fichero de configuración de Doxygen.

Para generar la documentación de su aplicación, colóquese en el directorio de su proyecto (src en esta práctica) y ejecute:
```
doxygen Doxyfile
```

Con el fichero Doxyfile que se suministra, la herramienta creará un subdirectorio doc en el directorio raíz de su proyecto en el que alojará toda la documentación generada. El directorio donde Doxygen genera su salida se especifica con la etiqueta OUTPUT_DIRECTORY (línea 61 del fichero Doxyfile suministrado). Con la configuración suministrada se generan 2 subdirectorios dentro de doc: html y latex. Si se coloca en el directorio doc/latex/ y ejecuta make el sistema "compila" el código LaTeX y genera un fichero refman.pdf que contiene la documentación generada. Observe en este caso el uso de make con un fichero Makefile no para compilar un programa sino para generar un fichero pdf a partir del código (texto) LaTeX. Si trabaja en su máquina virtual, traiga el fichero refman.pdf hacia su máquina local y visualice su contenido.

Si abre con un navegador el fichero doc/html/index.html accederá a la página principal de la documentación generada para el programa. En su máquina virtual no va a poder abrir un navegador para explorar los ficheros del directorio doc/html. Lo que ha de hacer es o bien generarlos en su instalación Linux local o bien traer el contenido del directorio doc desde su máquina virtual IaaS hacia su máquina local y en la máquina local abrir con un navegador el fichero doc/html/index.html.

Tal como se ha indicado, HTML o LaTeX son solo 2 de los formatos que permite generar Doxygen. Tanto HTML como LaTeX (también Markdown) son lo que se conoce como lenguajes de marcas. HTML es el lenguaje que se utiliza para componer los textos que se muestran en las páginas web. Latex es un sistema

de composición de textos que cuida el formato en especial en el ámbito de la tipografía y que es especialmente adecuado para textos de carácter científico. No se pretende aquí que profundice en conocer HTML o LaTeX.

La sección Documenting the code del manual de Doxygen indica cómo comentar el código fuente de modo que los comentarios sean procesados por Doxygen para incorporarlos a la documentación generada.

La guía Documenting C++ Code de documentación de código del proyecto LLST es la referencia que se adoptará en la asignatura para documentar el código de los programas que se desarrollen. Se utilizarán comentarios de tipo JavaDoc para comentarios de bloque:

```
/**
 * ... text comment ...
 */
```

JavaDoc es otro sistema de documentación ideado para Java y que también es muy popular. Doxygen soporta el uso de etiquetas "al estilo Javadoc" en el código.

Los bloques de comentarios multi-línea deben comenzar con

```
/**
```

y finalizar con

```
*/
```

Los comentarios de una única línea deben comenzar con `///`. Por consistencia no use las opciones

```
/*!
```

o

```
//!
```

permitidas en Doxygen.

Así el bloque de comentarios que debe preceder a cualquier función (o método) debiera tener una apariencia similar a esta:

```
/**
 * Sum numbers in a vector
 *
 * @param values Container whose values are summed
 * @return sum of `values`, or 0.0 if `values` is empty
 */
double SumValues(const std::vector<double>& values) {
  ...
}
```

En el ejemplo anterior `@param` y `@return` son etiquetas de tipo Javadoc. En Overview of supported JavaDoc style tags pueden consultarse este tipo de etiquetas.

El siguiente es un ejemplo (plantilla) de comentario de bloque que debería incluirse al comienzo de todos los ficheros (`*.cc`, `*.h`) de un proyecto de programación en el ámbito de esta asignatura:

```
/**
  * Universidad de La Laguna
  * Escuela Superior de Ingeniería y Tecnología
  * Grado en Ingeniería Informática
  * Informática Básica 2022-2023
  *
  * @author Albert Einstein aeinstein@ull.edu.es
  * @date Oct 12 2022
  * @brief El programa calcula la suma de todos los términos de valor par de la serie
  *        de Fibonacci que sean menores que un valor dado.
  *        Cada nuevo término de la serie se genera sumando los dos anteriores.
  *        Comenzando con 0 y 1, los primeros 10 términos serán: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
  * @bug There are no known bugs
  * @see https://www.cs.cmu.edu/~410/doc/doxygen.html
  */
```

Todo fichero debiera contener (etiqueta `@brief`) una breve descripción del contenido del fichero. Si fuera necesario se incluirá a continuación una descripción más detallada. Obviamente el comentario específico debiera particularizarse para cada caso concreto.

1.1.0.5 Material de estudio complementario

Estudie todo lo que se indica en el epígrafe `Functions` de la Guía de Estilo de Google y ponga en práctica todo lo que en ella se propone. Es normal si encuentra en ese epígrafe algún contenido que aún no ha estudiado: no se preocupe. Centre su atención en aquellos aspectos que ya conozca, para consolidarlos.

Estudie del `tutorial de referencia` en la asignatura los siguientes apartados:

- `Local variables`
- `Introduction to global variables`
- `Variable shadowing (name hiding)`
- `Scope, duration, and linkage summary`
- `Command line arguments`

1.1.0.6 Ejercicios

- Al realizar los ejercicios cree dentro de su repositorio de esta práctica un directorio diferente para cada uno de los ejercicios. Asigne a cada uno de esos directorios nombres significativos.
- Automatice la compilación del programa correspondiente a cada ejercicio con un fichero `Makefile` independiente para cada programa e inclúyalo en el correspondiente directorio.
- Haga que todos los programas tomen su entrada por la línea de comandos y en caso de que se ejecuten sin pasarles el número adecuado de parámetros impriman en pantalla un mensaje indicando el modo correcto de ejecutar el programa.
- El código de cada uno de los programas deberá organizarse de forma modular, es decir haciendo uso de funciones
- Cada función deberá realizar una única tarea y hacerlo correctamente
- El identificador de una función debe reflejar claramente lo que la función hace

1. Escriba un programa `function-example.cc` que incluya una función C++ que realice el cálculo de la siguiente función matemática de tres variables:

```
Public test cases
Input           Output
3 4 5           -0.349927
4.0 1.0 7.0      0.210819
1 2 3           -0.471405
3 2 10.0         0.8
```

1. Escriba un programa `change-case.cc`que tome como entrada una cadena de caracteres sin espacios e imprima como salida la misma cadena convirtiendo los caracteres que sean letras mayúsculas por minúsculas y viceversa. Los caracteres que no sean letras mayúsculas ni minúsculas deberán permanecer inalterados. Desarrolle su programa de forma que incluya como mínimo dos funciones y realice un diseño tal que el programa se organice en más de un fichero de código fuente (`*.cc`).

```
Public test cases
Input           Output
Abecedario      aBECEDARIO
PyThon          pYtHON
AlFa2022        aLfA2022
```

1. Desarrolle un programa `random_numbers.cc` que tome como entrada dos números naturales `n` y `m` tales que `n < m` y genere un número aleatorio real `r` en el intervalo `[n, m]`.

Desarrolle su programa de modo que incluya al menos dos funciones, y que sea una de ellas la que tome como parámetros los dos números introducidos por el usuario y devuelva como resultado el número aleatorio.

Para generar números aleatorios en C++ consulte la función `std::rand` por ejemplo en esta referencia.

Desarrolle un programa `random_numbers.cc` que tome como entrada dos números naturales `n` y `m` tales que n $<$ m y genere un número aleatorio real r en el intervalo [n, m]. Para generar números aleatorios en C++ consulte la función std::rand por ejemplo en esta referencia.

1. Estudie el programa `floating-point-arithmetics.cc` que se incluye en el directorio raíz del repositorio de esta práctica. Ese programa define una constante de tipo double, `kOneThird` cuyo valor (0.333) debiera coincidir con el de la fracción 1/3. Si se imprimen ambos valores por separado, aparentemente son iguales, sin embargo si se utiliza la expresión `1.0 / 3 == kOneThird` en una sentencia condicional, ésta resulta ser falsa. Este comportamiento se debe a la imprecisión que introduce la representación de los números en punto flotante en un ordenador. Para profundizar en este fenómeno, estudie el artículo Problem in comparing Floating point numbers and how to compare them correctly?

Desarrolle a continuación una función cuya definición sea: `bool AreEqual(const double number1, const double number2, const double epsilon = 1e-7);` que devuelva `true` o `false` dependiendo de si los números que se le pasan como parámetro son aproximadamente iguales o no. El tercer parámetro, que por defecto vale `1e-7` indica un valor muy pequeño que la función tomaría como margen de error a la hora de considerar si los números son o no iguales. Este artículo de *Tutorials Point* o Esta entrada de *StackOverflow* podrían resultarle útiles para desarrollar la función.

Pruebe el comportamiento de su función con diferentes pares de valores como entrada.

1. Escriba un programa `triangle-area.cc` que tome como entrada las longitudes `a`, `b` y `c`de los lados de un triángulo y calcule su área utilizando la Fórmula de Herón. Diseñe una función cuyo nombre sea *Area*, que implemente ese cálculo.

Los tres lados del triángulo, `a`, `b`, `c` deben satisfacer la desigualdad triangular: cada uno de los lados no puede ser más largo que la suma de los otros dos. Diseñe una función *IsAValidTriangle* que determine si los lados introducidos por el usuario forman un triángulo válido o no, y solo calcule su área en caso de ser válido

```
Public test cases
Input           Output
5 5 5            10.82
3 4 6             5.33
3.9 6.0 1.2      Not a valid Triangle
1.9 2 2           1.67
```

Chapter 2

README