UNIVERSIDAD
DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

Lenguaje y Ciencias de la Computación

TRABAJO FIN DE GRADO

Vídeos docentes para el aprendizaje de programación C/C++ en Ingeniería Industrial

Grado en Ingeniería de Organización Industrial

Autor: Sergio Fernández Casasola

Tutor: José Galindo Gómez

MÁLAGA, 13 de enero de 2023

DECLARACIÓN DE ORIGINALIDAD DEL PROYECTO/TRABAJO FIN DE GRADO

D. Sergio Fernández Casasola

DNI: 77182188F

Correo electrónico: sergiofernandezcasasola@gmail.com

Titulación: Grado en Ingeniería de Organización Industrial

Título del Proyecto/Trabajo: Vídeos docentes para el aprendizaje de programación C/C++ en Ingeniería Industrial

DECLARA BAJO SU RESPONSABILIDAD

Ser autor/a del texto entregado, inédito (no ha sido difundido por ningún medio, incluyendo internet) y original (no es copia ni adaptación de otra), no habiendo sido presentado anteriormente por mí ni por ningún otro autor o autora, ni en parte ni en su totalidad. Así mismo, se ha desarrollado respetando los derechos intelectuales de terceros, para lo cual se han indicado las citas necesarias en las páginas donde se usan, y sus fuentes originales se han incorporado en la bibliografía. Igualmente se han respetado los derechos de propiedad industrial o intelectual que pudiesen afectar a cualquier institución o empresa.

Para que así conste, firmo la presente declaración en Málaga, a 13 de enero de 2023

Fdo.: D/Dª _____

Vídeos docentes para el aprendizaje de programación C/C++ en Ingeniería Industrial

Sergio Fernández Casasola

Resumen

El objetivo del presente Trabajo de Fin de Grado es la continuación de un proyecto iniciado por el tutor de este trabajo, perteneciente al departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga. Consiste en la creación de material multimedia a modo de complemento para la enseñanza de la asignatura de Fundamentos de Informática de la Escuela de Ingenierías Industriales de Málaga. Este TFG se centra en la realización de vídeos docentes para mostrar la resolución de problemas de programación en Lenguaje C/C++, aunque también se han abarcado contenidos teóricos. En estos vídeos se mostrarán aclaraciones complementarias a los temas expuestos en clase, con un contenido entretenido y formativo relacionado con diversos campos de la ingeniería. Los temas que se tratan en la memoria se centrarán en la motivación de realizar contenido multimedia como complemento para la enseñanza de los alumnos y cómo se ha llevado a cabo dicho trabajo.

Palabras clave

Fundamentos de Informática, Programación, Lenguaje C/C++, Contenido multimedia, Vídeos, YouTube, Code::Blocks, PowerPoint, Audacity, Docencia online

ÍNDICE DE CONTENIDO

CAPÍTULO 1. INTRODUCCIÓN 11

1.1. Antecedentes 11

1.2. Objetivos generales del trabajo 12

1.3. Finalidad 13

1.4. Alcance 13

1.5. Plan de trabajo 14

1.6. Organización de la Memoria 14

CAPÍTULO 2. DOCENCIA ONLINE 17

2.1. Introducción a la docencia online: una opción para el aprendizaje en línea 17

2.2. Ventajas 18

2.3. Inconvenientes 20

2.4. Contenido multimedia para el aprendizaje de C/C++ 21

CAPÍTULO 3. METODOLOGÍA 25

3.1. Etapas de la elaboración de un vídeo 25

3.2. Investigación y documentación 30

3.3. Herramientas empleadas para la composición de nuestros vídeos 31

3.3.1. PowerPoint 32

3.3.2. Audacity 33

3.3.3. Code::Blocks 39

CAPÍTULO 4. MATERIAL AUDIOVISUAL CREADO 41

4.1. Material audiovisual ya existente en el canal "*Aprende conmigo*" 41

4.2. Tema 3. Introducción a C/C++ 42

4.2.1. Vídeo: Operaciones básicas del tipo string en C/C++ 44

4.2.2. Vídeo: Operadores ++ y −− en C/C++ (preincremento, preincremento, postincremento, predecremento y postdecremento) 46

4.2.3. Vídeo: Operaciones simples entre enteros y reales (tipos de datos int, float, double y long double) 47

4.2.4. Vídeo: Programa C/C++ simple sobre el tipo string 48

4.2.5. Vídeo: Programa C/C++ para ver el código de un carácter (molde int) y su tamaño en bytes (operador sizeof) 49

4.2.6. Vídeo: Programa C/C++ para resolver una ecuación de segundo grado 50

4.3. Tema 4. Estructuras de Control 51

4.3.1. Vídeo: Programa C/C++ para contar cuántas veces aparece un carácter en una frase 52

4.3.2. Vídeo: Programa C/C++ para calcular las raíces de una función matemática por el método de Newton-Raphson 54
4.3.3. Vídeo: Dibujar un triángulo de asteriscos (relleno y hueco)..... 55
4.3.4. Vídeo: Programa C/C+ para analizar un Circuito Eléctrico simple con 3 resistencias en serie o en paralelo 57
4.4. Tema 5. Funciones 59
4.4.1. Vídeo: Programa C/C++ para calcular la distancia euclídea entre 2 puntos 2D, con funciones 60
4.4.2. Vídeo: Programa C/C++ para comprobar la Conjetura de Goldbach 62
4.5. Tema 6. Datos Estructurados 64
4.5.1. Vídeo: Programa C/C++ sencillo para hallar el Centro de masas de un conjunto de partículas en 3D 64
4.5.2. Vídeo: Criba de Eratóstenes en lenguaje C/C++ para hallar números primos. 66
CAPÍTULO 5. CONCLUSIONES Y TRABAJOS FUTUROS..... 69
5.1. Conclusiones 69
5.2. Aprendizaje 70
5.3. Objetivos cumplidos..... 71
5.4. Ventajas y desventajas encontradas 71
5.5. Posibles mejoras y trabajos futuros 74
REFERENCIAS 75
ANEXO 1: VÍDEOS CREADOS Y SUS ENLACES 77
A.1. Tema 3..... 77
A.2. Tema 4..... 77
A.3. Tema 5..... 78
A.4. Tema 6..... 78

ÍNDICE DE ILUSTRACIONES

Figura 3.1: Portada vídeo: Conjetura de Goldbach. Fuente: Elaboración propia 27
Figura 3.2: Despedida vídeo: Conjetura de Goldbach. Fuente: Elaboración propia 28
Figura 3.3: Vídeos sugeridos por YouTube en los últimos segundos. Fuente: Elaboración propia 29
Figura 3.4: Reducción de ruido en Audacity. Fuente: Elaboración propia 35
Figura 3.5: Compresor en Audacity. Fuente: Elaboración propia 36
Figura 3.6: Ecualizador de curva de filtro en Audacity. Fuente: Elaboración propia 37
Figura 3.7: Limitador en Audacity. Fuente: Elaboración propia 38
Figura 3.8: Normalizar en Audacity. Fuente: Elaboración propia 38
Figura 3.9: Amplitud de señal de una grabación antes y después de ser editada. Fuente: Elaboración propia 39
Figura 4.1: Una imagen de ejemplo del vídeo “Operaciones básicas del tipo string”. Fuente: Anexo 1. ..45
Figura 4.2: Una imagen de ejemplo del vídeo “Operadores de incremento y decremento”. Fuente: Anexo 1. 46
Figura 4.3: Una imagen de ejemplo del vídeo: “Operaciones simples entre enteros y relaes”. Fuente: Anexo 1. 47
Figura 4.4. Una imagen de ejemplo del vídeo: “Programa C/C++ simple sobre el tipo de dato string”. Fuente: Anexo 1. 48
Figura 4.5 Una imagen de ejemplo del vídeo: “Programa C/C++ para ver el código de un carácter y su tamaño en bytes”. Fuente: Anexo 1 50
Figura 4.6: Una imagen de ejemplo del vídeo: “Programa C/C++ para resolver una ecuación de segundo grado”. Fuente: Anexo 1. 50
Figura 4.7: Una imagen de ejemplo del vídeo: “Programa C/C++ para contar cuántas veces aparece un carácter en una frase”. Fuente: Anexo 1. 53
Figura 4.8: Una imagen de ejemplo del vídeo: “Programa C/C++ para calcular las raíces de una función matemática por el método de Newton-Raphson”. Fuente: Anexo 1. 54
Figura 4.9: Una imagen de ejemplo del vídeo: “Dibujar un triángulo de asteriscos (relleno y hueco). Fuente: Anexo 1. 56
Figura 4.10: Una imagen de ejemplo del vídeo: “Programa C/C++ para analizar un Circuito Eléctrico simple con 3 resistencias en serie o en paralelo. Fuente: Anexo 1. 57
Figura 4.11: Una imagen de ejemplo del vídeo: “Programa C/C++ para calcular la distancia euclídea entre 2 puntos 2D con funciones”. Fuente: Anexo 1. 60
Figura 4.12: Una imagen de ejemplo del vídeo: “Programa C/C++ para comprobar la Conjetura de Goldbach”. Fuente: Anexo 1. 62
Figura 4.13: Una imagen de ejemplo del vídeo: “Programa C/C++ para hallar el Centro de masas de un conjunto de partículas en 3D”. Fuente: Anexo 1. 65

Figura 4.14: Una imagen de ejemplo del vídeo: “Criba de Eratóstenes en lenguaje C/C++ para hallar números primos”. Fuente: Anexo 1 67

Figura 5.1: Ejemplo de diapositiva de un vídeo en PowerPoint. Fuente: Elaboración propia 73

ÍNDICE DE TABLAS

Tabla 4.1: Listas de reproducción del canal “Aprende conmigo” antes del presente TFG. Nota: observe que no había vídeos del Tema 3. Fuente: [11] 42

Tabla 4.2: Listas de reproducción del canal “Aprende conmigo” después del presente TFG. Se marcan en negrita los temas que se han incorporado. Fuente: [11] 42

Tabla 4.3: Declaración de variables en C++. Fuente: Elaboración propia 43

Tabla 4.4: Asignación de un valor a una variable en C++. Fuente: Elaboración propia 43

Tabla 4.5: Ejemplo de algunos operadores en C++. Fuente: Elaboración propia 44

Tabla 5.1: Relación tareas-tiempo invertido en la realización del presente TFG. Fuente: Elaboración propia 70

Tabla 5.2: Resumen de los vídeos creados para cada tema con su duración en minutos. Fuente: Elaboración propia 72

CAPÍTULO 1.

INTRODUCCIÓN

En este capítulo se abordará los principales motivos que han motivado la creación de este Trabajo de Fin de Grado (TFG). Para ello, se especificará el objetivo del trabajo y se presentará un marco conceptual que permita entender el contexto en el que se enmarca. También se analizará el alcance que supone utilizar una plataforma de difusión como es YouTube [21]. Por último se expondrá el plan de trabajo que se ha seguido para llevar a cabo el presente trabajo.

Así pues, este documento aborda la creación de vídeos de programación de lenguaje C o C++ como una herramienta de enseñanza efectiva. Se analizarán diferentes plataformas y técnicas de grabación y edición de vídeo, así como nuestra opinión en cuanto a las mejores prácticas experimentadas para hacer tutoriales de programación atractivos y fáciles de seguir. Además, se explorará una nueva tendencia que ha cobrado especial importancia hoy en día: la educación en línea.

1.1. Antecedentes

La programación ha sido una de las causantes del crecimiento tecnológico producido en las últimas décadas. No es de extrañar que las asignaturas relacionadas con la programación cobren mayor relevancia. Es lógico, por tanto, que existan asignaturas dentro de los planes de estudios de ingeniería con contenidos relacionados a la programación en la formación de cualquier ingeniero o ingeniera. Este es un desafío en la enseñanza porque algunos conceptos son difíciles de entender y requieren de una dedicación significativa por parte de los estudiantes.

Este trabajo busca aclarar, revelar, guiar y profundizar algunos de dichos conceptos que puedan suponer una dificultad para aquellos estudiantes que no disponen del tiempo o los conocimientos matemáticos suficientes para analizar y comprender con la rapidez que se precisa en las clases presenciales. Para facilitar esta tarea el tutor de este trabajo viene desarrollando contenido audiovisual que se ofrece a los alumnos a lo largo del curso. La propia Universidad de Málaga ha desarrollado varios PIE (Proyectos de Innovación Educativa) que van en esta misma línea, precisamente este TFG se ha propuesto para formar parte de este con el código: PIE22-218.

Un PIE es una iniciativa que busca mejorar la calidad de la educación en la institución a través de la implementación de nuevas estrategias, metodologías, herramientas y tecnologías. Estos pueden ser llevados a cabo por profesores, estudiantes, investigadores o cualquier miembro de la comunidad universitaria y pueden abarcar una amplia gama de temas y áreas de conocimiento. El objetivo de estos proyectos es promover la creatividad, la investigación y la excelencia en el ámbito de la educación, y pueden ser financiados por diversas fuentes.

Hoy en día, los estudiantes están acostumbrados a recibir la mayor parte de la información de forma visual. Como es de esperar, cuando se trata de consultar dudas o reforzar conocimientos, una de las primeras opciones es recurrir a Internet, siendo cada vez más frecuencia la búsqueda de contenido en forma de material audiovisual (vídeos, podcast, presentaciones…).

La cantidad de plataformas web dedicadas a compartir vídeos ha crecido exponencialmente en los últimos años, especialmente YouTube que cuenta con casi tres mil millones de cuentas registradas [4]. La cantidad y variedad de vídeos mejora la facilidad de consumo, convirtiéndolo en uno de los medios preferidos por los estudiantes. Si bien en YouTube se pueden encontrar vídeos relacionados con la programación, surge el problema de la fiabilidad de la información, encontrándonos vídeos que facilitan conceptos explicados de forma errónea o de forma engorrosa y de larga duración. De ahí, nuestra propuesta de ofrecer un contenido revisado y fiel a los contenidos teóricos de la propia asignatura.

Algunos de los vídeos en el canal de YouTube de este proyecto[1] acumulan miles de visitas y comentarios muy positivos desde distintos puntos del globo. Esto da idea de la utilidad y alcance de este trabajo.

1.2. Objetivos generales del trabajo

El objetivo principal de este trabajo es ofrecer al alumnado contenido audiovisual, intentando que sea de forma entretenida y formativa a través de la resolución de problemas de programación en C/C++ relacionados con diversos campos de la ingeniería industrial. En algunos vídeos también se explicarán o se repasarán conceptos teóricos.

El estilo y formato de los vídeos se mantendrá igual que los ya ofrecidos en el canal *"Aprende conmigo"* disponible en YouTube [11]. Para ello, se hará uso de los mismos programas informáticos:

- *PowerPoint:* es el programa utilizado para la elaboración del contenido y su producción.
- *Audacity:* este programa permite grabar y editar el audio.
- *Code::Blocks:* es el entorno de desarrollo integrado de código abierto utilizado para la compilación de programas en lenguaje C/C++.

Cabe remarcar la necesidad de mantener el formato audiovisual para que no se aprecien cambios significativos en la forma de edición y lograr un estilo homogéneo que se pueda reproducir sin importar quien elabore el vídeo.

[1] *Bucles anidados y ejemplos en Lenguaje C - YouTube*. (s. f.). Recuperado 6 de enero de 2023, de https://www.youtube.com/watch?v=mUdDd1jGg8U

La CPU: UC, ALU y las fases de ejecución de una instrucción - YouTube. (s. f.). Recuperado 6 de enero de 2023, de https://www.youtube.com/watch?v=-URf73z9tKY

1.3. Finalidad

La finalidad de crear contenido multimedia para el alumnado es proporcionar una fuente fiable de información a los mismos de forma eficaz y atractiva que sirva como apoyo a los conceptos explicados y desarrollados en clase. Los vídeos pueden ser una valiosa fuente de información para los estudiantes, ya que proporcionan una explicación visual y detallada de conceptos y técnicas de programación que profundizan los conceptos vistos en la propia asignatura.

Estos vídeos también pueden ser útiles para los alumnos que tienen dificultades para comprender conceptos a través de la lectura o las explicaciones en clase. Al mostrar y explicar cómo funcionan las cosas de forma visual y práctica, los vídeos pueden facilitar el aprendizaje y hacerlo más ameno para los alumnos.

Además, estos contenidos audiovisuales creados pueden ser una valiosa fuente de información para los estudiantes que quieran aprender de forma autodidacta o mejorar sus conocimientos de programación. Los vídeos explican paso a paso cómo resolver diferentes escenarios o problemas de programación y algunos conceptos teóricos, lo que los convierte en una herramienta útil para quienes desean aprender de forma autónoma.

1.4. Alcance

El alcance de crear vídeos de programación en YouTube puede variar enormemente según varios factores, como pueden ser la calidad del contenido, la efectividad de la promoción del vídeo y la audiencia objetivo. Pero, en general, YouTube es una plataforma muy popular que puede brindar un alcance bastante significativo a los vídeos de programación.

Una de las principales ventajas de utilizar YouTube como plataforma de difusión es su accesibilidad y usabilidad. Cualquiera puede consumir contenido audiovisual en la plataforma, así como crearse una cuenta y comenzar a subir vídeos, lo que significa que hay una amplia audiencia potencial para los vídeos de programación. Además, YouTube tiene un sistema de sugerencias y recomendaciones de vídeos que puede ayudar a llegar a más personas.

Otro factor que afecta al alcance de los vídeos de programación en YouTube es la calidad del contenido. Es más probable que los vídeos que ofrecen información útil y bien presentada sean más compartidos y por ende, atraigan a una audiencia más amplia. Por otro lado, los vídeos de baja calidad o poco interesantes pueden tener un alcance limitado.

Por último, la promoción que reciba un vídeo también afecta enormemente al alcance de este. Por ejemplo, compartir un vídeo en redes sociales u otros sitios web puede ayudar a que más personas vean el vídeo en YouTube.

1.5. Plan de trabajo

Para poder llevar a cabo este trabajo ha sido necesario cumplir ciertas tareas y actividades que se describen a continuación:

1. Estudio del contenido teórico y práctico de la asignatura Fundamentos de Informática y del material ya disponible en el canal de YouTube.

2. Lectura de documentación relacionada con la programación en C/C++ para poder afrontar el presente trabajo con una base de conocimientos suficientemente sólida.

3. Redacción del enunciado de los problemas específicos para cada Tema de la asignatura.

4. Elaboración del contenido audiovisual de carácter teórico y práctico.

5. Edición de los vídeos de carácter teórico y práctico.

6. Proceso iterativo de corrección y edición.

7. Subida de los vídeos al canal de YouTube *"Aprende conmigo"*, con fines docentes.

8. Redacción de la memoria del trabajo, así como la presentación para la defensa de este.

1.6. Organización de la Memoria

La organización del resto del presente documento se estructura de la siguiente manera:

- En el Capítulo 2 se aborda el tema de la docencia online, exponiendo sus ventajas y desventajas, así como los motivos de su repentino crecimiento debido a la pandemia del COVID-19.

- En el Capítulo 3 se expondrá y desarrollará la metodología que se ha seguido para llevar a cabo el presente trabajo y lograr los objetivos propuestos. También, se profundizará en las herramientas utilizadas, así como pequeñas observaciones de cara a futuros trabajos en la misma línea.

- En el Capítulo 4 se detallarán los vídeos creados para el presente trabajo. A modo de contexto se expondrá un breve resumen del temario de la asignatura, la cual ha servido de base para enfocar el contenido de los vídeos, así como un índice general de los vídeos ya existentes previamente en el canal de YouTube. También, se expondrán todos los vídeos creados con un pequeño resumen de los conceptos trabajados, así como el código utilizado.

- En el Capítulo 5 se presentan las conclusiones recabadas. Se dará una visión del trabajo que ha supuesto, así como de las experiencias vividas, lo que ha aportado tanto personalmente como al colectivo al que va dirigido, ventajas y desventajas encontradas y posibles mejoras para seguir expandiendo el contenido del canal de YouTube "*Aprende conmigo"*.

Al final de la memoria se encuentran las referencias bibliográficas consultadas para la elaboración del presente documento, así como del contenido audiovisual creado. También, se incluye un anexo donde se encuentran los enlaces al canal de YouTube *"Aprende conmigo"* y los vídeos creados en cuestión.

CAPÍTULO 2.

DOCENCIA ONLINE

En este capítulo, se abordará el tema de la docencia online y su reciente crecimiento impulsado por la pandemia del COVID-19 [6]. Además, se examinarán los aspectos positivos y negativos de la educación online, así como algunas de las consideraciones para tener en cuenta al elegir este tipo de formación. También se hará una breve comparativa respecto a la actual docencia presencial.

2.1. Introducción a la docencia online: una opción para el aprendizaje en línea

La importancia del aprendizaje en línea ha aumentado considerablemente en los últimos años, debido en parte al auge de la demanda de flexibilidad y accesibilidad en el aprendizaje a distancia. Esta forma de enseñanza ha demostrado ser una opción eficaz y cada vez más popular para muchas instituciones educativas y estudiantes de todo el mundo. La educación en línea tiene varias ventajas, como la comodidad, la flexibilidad, la accesibilidad y el ahorro de costes y transportes.

Sin embargo, también existen desafíos asociados con la docencia online y requiere una planificación e implementación cuidadosas para tener éxito. Los docentes deben adaptar sus métodos de enseñanza y aprender a utilizar diversas herramientas tecnológicas para impartir docencia a través de Internet. También deben estar preparados para explorar nuevos métodos y estrategias de enseñanza para aprovechar al máximo las oportunidades que ofrece la educación en línea.

Durante la pandemia de COVID-19 [6], la importancia de la educación en línea ha aumentado debido a la obligación que hubo de cerrar las instituciones educativas y prohibir las reuniones públicas, donde se congregaban grandes cantidades de personas para evitar la propagación del virus. La educación online ha permitido que muchas instituciones educativas continuaran ofreciendo clases y permitiendo que los estudiantes completaran su educación sin interrupción. La educación online abarca tanto actividades síncronas como asíncronas. En las primeras se incluyen sesiones de trabajo en las que hay que conectarse a Internet en un horario concreto, tales como clases online en las que el profesor explica y los alumnos pueden interactuar con él, preguntando dudas, por ejemplo. Las actividades asíncronas por su parte, son aquellas que pueden emplearse en cualquier momento, sin necesidad de sincronizarse con nadie. Los vídeos entran en esta categoría y tienen el inconveniente de que no se puede interactuar en directo, aunque siempre los alumnos pueden tener otros canales para plantear sus dudas (mail, foros…).

Además, la docencia online ha otorgado una mayor flexibilidad y accesibilidad a los estudiantes durante la pandemia. Muchos estudiantes se han encontrado con problemas para asistir a clases presenciales debido a la necesidad de cuidar de familiares enfermos, cumplir con compromisos laborales o simplemente por el miedo a contagiarse del virus. La docencia online ha permitido a estos estudiantes continuar su educación desde la comodidad y seguridad de sus hogares.

La docencia online ha ayudado a reducir la brecha digital y garantizar el acceso a la educación para todos los estudiantes, independientemente de su ubicación geográfica o recursos financieros. Muchos estudiantes de áreas remotas o que poseen bajos ingresos no tienen acceso a la misma calidad de educación que los estudiantes en áreas más urbanas o de mayores ingresos. La docencia online les ha permitido a estos estudiantes tener acceso a clases de alta calidad y recursos de aprendizaje sin tener que viajar a una institución educativa física.

En resumen, la docencia online durante la pandemia de COVID-19 ha demostrado ser una valiosa opción para continuar brindando una educación de calidad y garantizar la flexibilidad y accesibilidad necesarias para los estudiantes. Ahora que la pandemia parece estar controlada, en el ámbito académico se está volviendo a optar por un modelo educativo presencial, este hecho no quita que la docencia online siga siendo una opción considerable para las instituciones educativas y los estudiantes en todo el mundo.

Por otro lado, es importante recordar que la educación online no es simplemente una versión en línea de la enseñanza presencial. Los docentes deben ser creativos y pensar en formas innovadoras de involucrar a los estudiantes y promover el aprendizaje a través de la tecnología. Esto puede incluir el uso de foros en línea, videoconferencias, juegos educativos y otras actividades interactivas.

Otro aspecto que destacar de la docencia online es la comunicación y el apoyo hacia los estudiantes. Para garantizar el éxito académico, es importante establecer una línea de comunicación clara y constante con los estudiantes y proporcionar una amplia variedad de recursos y servicios de apoyo. Esto puede incluir tutorías en línea, materiales de estudio y asesoramiento académico. Hay que tener especial cuidado a la hora de crear un ambiente de aprendizaje virtual efectivo. Para ello, es importante asegurarse de que el material del curso sea interesante y relevante para los estudiantes y que haya suficientes oportunidades para la discusión y el debate en línea.

Resumidamente, la docencia online puede ser una alternativa efectiva y conveniente para el aprendizaje a distancia, siempre y cuando se planifique y se implemente de manera cuidadosa. Los docentes deben ser flexibles y creativos al adaptar sus métodos de enseñanza y utilizar diferentes herramientas tecnológicas para involucrar y apoyar a los estudiantes. De esta manera, se pueden crear experiencias de aprendizaje enriquecedoras y efectivas a través de la educación en línea.

2.2. Ventajas

A modo de esquema se enumeran una lista de ventajas que se comentarán en esta sección:

- Flexibilidad
- Personalización
- Accesibilidad
- Variedad de recursos y herramientas
- Amplía variedad de programas educativos
- Eficiencia
- Comodidad
- Ahorro de costes

Una de las principales ventajas de la docencia online es la flexibilidad que ofrece tanto a los docentes como a los estudiantes. Los docentes pueden dar clases desde cualquier lugar donde haya una conexión a Internet, lo que les permite enseñar a estudiantes de todo el mundo. Además, Los estudiantes pueden acceder al material del curso y completar las actividades y tareas a su propio ritmo y en su propio horario, siempre y cuando cumplan con las fechas límite establecidas. Esta flexibilidad es especialmente beneficiosa para quienes tienen compromisos familiares o laborales que les impiden asistir a clases presenciales. Los estudiantes pueden acceder al material del curso y completar las actividades y tareas a su propio ritmo y en su propio horario, siempre y cuando cumplan con las fechas límite establecidas.

Otro aspecto positivo es la variedad de recursos y herramientas que están disponibles para apoyar el aprendizaje. Los docentes pueden usar plataformas en línea como Moodle [15] Blackboard [5] para subir materiales de estudio y aprendizaje, publicar tareas y calificaciones, e incluso comunicarse con los estudiantes. También pueden usar videoconferencias y foros en línea para comunicarse e interaccionar con los estudiantes y fomentar el debate y el intercambio de ideas. Estas herramientas son particularmente útiles para promover el aprendizaje colaborativo y la participación activa de los estudiantes. Hay que tener especial cuidado a la hora de seleccionar la plataforma adecuada. Existen muchas opciones disponibles, cada una con sus propias características y beneficios. Es importante elegir una plataforma que sea fácil de usar tanto para el docente como para los alumnos y que brinde todas las herramientas y recursos necesarios para impartir y participar en las clases en línea.

Los cursos en línea ofrecen a menudo una gama más amplia de opciones que los programas tradicionales en la universidad, lo que permite a los estudiantes adaptar su educación a sus intereses específicos y objetivos profesionales. Además, los programas en línea a menudo ofrecen cursos acelerados que permiten a los estudiantes completar su título más rápido que en un programa tradicional. Esto supone una importancia considerable en el sector tecnológico, pues hoy en día cada vez más las empresas precisas de profesionales más especializados en áreas concretas, por lo que estos cursos con contenidos más focalizados y de menor duración suponen una enorme ventaja frente a los cursos tradicionales que oferta la universidad. Una plataforma que capitaliza casi por completo los cursos online es Udemy [18], en esta se puede encontrar cursos de prácticamente cualquier área, con una amplia variedad dentro de la misma.

Destacar que incluso las propias instituciones ofrecen una gran variedad de programas y cursos en línea. Por ejemplo, la Universidad Nacional de Educación a Distancia (UNED) [19] es una universidad pública española con sede en Madrid. Fue fundada en 1972 y es la mayor universidad a distancia de España y una de las más grandes del mundo. Esta ofrece una amplia variedad de programas de licenciatura, máster y doctorado a través de su plataforma de educación a distancia. La UNED también tiene una red de centros de estudio repartidos por toda España, donde los estudiantes pueden asistir a clases presenciales y recibir apoyo académico. Aunque si bien es cierto que la mayoría de los programas que ofrece la UNED son a distancia, a veces requiere una presencialidad para realizar ciertas pruebas o prácticas. Por lo que no podríamos considerarlo una educación en línea total sino parcial.

Otra ventaja considerable de la educación en línea es su eficiencia. Los profesores pueden usar una gran variedad de herramientas tecnológicas para enseñar y proporcionar materiales de estudio, ahorrando tiempo y esfuerzo tanto para los maestros como para los estudiantes. Además, muchas plataformas de educación en línea ofrecen una amplia variedad de recursos y servicios de

apoyo para garantizar el éxito académico de los estudiantes, como tutorías en línea, materiales de estudio y asesoramiento académico.

Este tipo de docencia puede ser muy cómoda para los estudiantes, esto se debe a que al disponer de todo el material y poder acceder a las clases desde Internet se relega la necesidad de desplazarse físicamente a unas instalaciones para asistir a clases. Además, es muy común disponer de las clases grabadas, por lo que el propio alumno puede marcar el ritmo del aprendizaje, pudiendo establecer su propio horario a conveniencia y optimizar cuando consumir dicho material para sacarle el máximo rendimiento al aprendizaje.

Otra ventaja que tener en consideración es el ahorro de costes. Los cursos y programas en línea suelen tener un coste menor que sus homólogos presenciales, ya que no requieren que los estudiantes paguen alojamiento, transporte u otros gastos asociados con el estudio en la universidad. Además, muchas instituciones ya ofrecen programas de educación en línea a precios más asequibles que sus homólogos presenciales. Esto puede hacer que la educación en línea sea una opción más asequible para muchos estudiantes. Precisamente, los cursos que se encuentran disponibles en Udemy son generalmente más asequibles que los cursos tradicionales, y muchos de ellos ofrecen una versión de prueba gratuita. Simplemente el hecho de poder probar un servicio para comprobar que se adapte a las necesidades del consumidor supone una enorme diferenciación.

2.3. Inconvenientes

La docencia online y la docencia presencial son dos modalidades de enseñanza que tienen sus propias ventajas y desventajas. La elección de una u otra dependerá de las necesidades y preferencias de las instituciones educativas y de los estudiantes.

Sin embargo, no todos son ventajas pues la educación en línea también presenta algunos desafíos. Puede ser mucho más complicado involucrar y motivar a los estudiantes a través de la tecnología y además, puede haber menos oportunidades para la interacción cara a cara con los profesores y compañeros de clase. Precisamente, una desventaja potencial es que los cursos en línea pueden no proporcionar el mismo nivel de interacción y apoyo que los cursos tradicionales. Los estudiantes pueden perder la oportunidad de hacer preguntas y recibir comentarios, lo que puede ser importante para su aprendizaje y desarrollo. Además, algunos estudiantes pueden tener problemas con la automotivación y la disciplina necesarias para tener éxito en un curso en línea, ya que no tienen el mismo nivel de estructura y apoyo que en un aula tradicional.

En cambio, en la docencia presencial si existe una interacción cara a cara con los profesores y compañeros de clase, lo que puede ser particularmente valioso para algunos estudiantes. Además, la docencia presencial puede ser más apropiada para ciertos tipos de materias o asignaturas que requieren de una mayor participación y discusión personal. Sin embargo, la enseñanza presencial también presenta algunos desafíos, como la necesidad de asistir a clases en un horario específico y en una ubicación física determinada.

Otra desventaja que presenta la docencia online es que los alumnos tienen mayor facilidad para despistarse, al encontrarse en un entorno como puede ser su casa, simplemente el hecho de poder hacer cualquier otra actividad ya supone una distracción, como puede ser usar el móvil, ir a beber agua, mirar contenido en Internet, e incluso el ruido que pueda haber en la calle ya supone una posible

fuente de distracción. Además, la falta de motivación cobra especial importancia en estos casos, pues alguien que este desmotivado lo tiene más fácil para distraerse.

La falta de gestión del tiempo y saber cómo estructurarse puede suponer una enorme diferencia en este tipo de enseñanza, ya que si el alumno no es bueno gestionando y organizando su tiempo de manera efectiva, puede ser difícil seguir el ritmo de un programa de educación en línea.

Los problemas técnicos también suponen un gran inconveniente, al final es necesario disponer de un ordenador con conexión a Internet para poder seguir el aprendizaje, por lo que cualquier problema relacionado con ambos o incluso un problema externo que pueda afectar a la propia plataforma implica no poder consumir el contenido de aprendizaje.

Por último, es posible que algunos estudiantes requieran de mayor esfuerzo para poder mantenerse al día y aprovechar al máximo el programa educativo, esto se puede deber a diversos factores como puede ser las dificultades que pueda tener para comprender el contenido o la falta de apoyo.

En general, como hemos visto la educación en línea puede ser una opción cómoda y flexible para los estudiantes, especialmente para los que tienen otros compromisos o buscan una forma más asequible de obtener un título. Sin embargo, es importante que los estudiantes consideren cuidadosamente que estilo de aprendizaje y necesidades se adapta mejor antes de matricularse en un programa en línea, para asegurarse que puedan tener éxito y sacar el máximo provecho de su educación.

2.4. Contenido multimedia para el aprendizaje de C/C++

La era digital ha revolucionado la forma en que adquirimos conocimientos y ha aumentado la disponibilidad de información a la que podemos acceder gracias a Internet, pudiendo consultar infinidad de páginas web, libros, vídeos o incluso foros de debate. Estas son una fuente de conocimiento valiosas para aprender, ya que dentro de las mismas, tanto la forma de plasmar la información como el contenido pueden ser muy diversas. Especialmente el contenido multimedia, ofrece una experiencia visual y auditiva que puede ser más efectiva para algunas personas que un libro de texto o clases presenciales impartidas por un tutor.

Tanto el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga como el Departamento de Ingeniería Informática de la Universidad de Cádiz son conscientes de este nuevo cambio en el paradigma de la enseñanza. Es por ello, que están trabajando en crear contenido multimedia sobre programación en lenguaje C/C++. En el artículo *"Multimedia system for self-learning C/C++ programming language"* [12] presentado por investigadores de ambos departamentos se expone que desde su experiencia como docentes, aprender programación en cuatro meses es una tarea ardua, pero así lo exigen los planes de estudio de muchas universidades en España. Dicho aprendizaje se enfrenta a algunas desventajas ya comentadas previamente en este documento acerca de la docencia presencial, como puede ser la flexibilidad, ya que hay una gran cantidad de estudiantes que no pueden asistir a clases con regularidad, o que incluso si asisten se encuentren con una clase que no pueden seguir debido a su falta de conocimiento causado por su ausencia. Teniendo en cuenta estas adversidades, los departamentos de ambas universidades decidieron crear contenido

multimedia para apoyar a los estudiantes y ofrecerles un complemento que ayude a su aprendizaje y afianzar los conocimientos ya adquiridos durante las clases.

El contenido multimedia descrito presenta las siguientes características:

- **Ritmo de aprendizaje**. Al ser un contenido multimedia, el alumno puede parar y volver a empezar el vídeo siempre que quiera, así como repetirlos tantas veces como desee.

- **Animaciones y sonido**. Existen animaciones y sonidos que hacen la experiencia mucho más atractiva y dinámica. Esto, unido a la voz en off del locutor, busca centrar y llamar la atención del alumno. Por ejemplo, de forma visual el alumno puede distinguir como cambia el valor de una variable en diferentes sentencias, mientras se ejecuta el programa.

- **Explicaciones paso a paso**. Se desarrollan y explican conceptos teóricos y prácticos, que también se enlazan con ejemplos cuya complejidad aumenta en función del temario.

- **Programas de ejemplo**. Se incluyen programas completos, fragmentos o funciones de ejemplo, explicando su ejecución paso a paso. Al final de cada vídeo se deja para el espectador varios programas con menos explicaciones, invitando al espectador a comprobarlos, comprenderlos y modificarlos.

- **Conceptos complejos**. Los conceptos más complejos se explican de varias formas porque se ha detectado que ciertos alumnos presentan dificultad a la hora de entender y comprender estos conceptos. De esta forma, explicar un mismo concepto de diferentes maneras ayuda al alumno a terminar de entender y comprender los conceptos más complejos en cuanto a programación.

Este TFG persigue las ambiciones, objetivos y metodologías expuestos en el artículo [12]. Por ello, para añadir más contenido multimedia se decidió crear una nueva serie de vídeos centrados en resolver problemas de programación. La premisa que siguen estos vídeos es la siguiente: se plantea un problema al comienzo del vídeo acorde al temario correspondiente de la asignatura, luego se explican los conceptos de programación que se utilizarán para resolver el problema y por último se explica cada parte del código utilizado tanto a nivel conceptual como técnico. Al final del vídeo siempre se muestra una diapositiva con todo el código que resuelve el ejercicio, ayudando así al espectador a tener una visión global de todo el código sin la necesidad de tener que ir saltando distintas partes del vídeo.

Resulta interesante destacar una encuesta realizada a los estudiantes acerca de la efectividad de los vídeos ya propuestos y publicados en el canal de YouTube [12]. Dicha encuesta se presenta a los estudiantes una vez por clase y curso, y se ruega a los estudiantes que la rellenen cada vez que consuman el contenido multimedia puesto a su disposición. Los estudiantes son informados de que rellenar esta encuesta es totalmente anónimo y opcional, pero que resulta de enorme ayuda para mejorar tanto el material como el contenido.

Alguna de las preguntas y resultados más interesantes de esta encuesta son:

1. ¿Considera que este formato multimedia es interesante y útil para aprender?
   - Respuesta media: 4,6 sobre 5 (desviación típica: 0,3).

2. ¿Le parecen adecuadas las explicaciones?
   - Respuesta media: 4 de 5 (desviación típica: 0,6).

3. ¿Le parecen adecuados los ejemplos utilizados?
   - Respuesta media: 4,2 de 5 (desviación típica: 0,2).

4. ¿Ha repetido o va a repetir todas o algunas partes de las explicaciones?
   - Tres respuestas: No (0%); Todas (16,7%); Algunas partes (83,7%).

5. ¿Cree que es conveniente utilizar este material en clase o es mejor dejarlo solo para el estudio individual? Cinco respuestas:
   a) Prefiero solo las explicaciones del profesor en clase (33,3%);
   b) En clase se deben mezclar las explicaciones del profesor con poco uso de material multimedia (50%);
   c) La opción ideal es combinar ambos métodos por igual (16,7%);
   d) La opción ideal es utilizar principalmente el material multimedia, y que el profesor solo respondiera a las preguntas o añadiera algunos comentarios (0%);
   e) Prefiero solo material multimedia en clase (0%);

6. Valore la rapidez de las explicaciones, en general:
   - Tres respuestas: Demasiado rápido (16,7%); Buen ritmo (83,3%); Demasiado lento (0%).

7. ¿Le gustaría tener muchos temas de todas las asignaturas en este formato?
   - Respuesta media: 4,3 de 5 (desviación típica: 0,3).

Las encuestas realizadas a los alumnos revelan en su mayoría que estos agradecen y utilizan dicho material para sus estudios. También reflejan que no les gusta que se utilice este material en clase, dejando este material como un complemento externo para consultar fuera de clase, relegando la tarea del docente a resolver dudas. Como ya se ha comentado, una de las ventajas de la docencia presencial es la efectividad que tiene a la hora de la interacción profesor-alumno, permitiendo por parte de los alumnos levantar la mano y preguntar en cualquier momento una duda concreta que les pueda surgir.

En general, el contenido de estas encuestas ayuda enormemente de cara a mejorar el contenido y crear nuevo material enfocado en las necesidades de los alumnos. Como se puede apreciar, el resultado de la encuesta es considerablemente satisfactorio, por ello hay que seguir trabajando para alcanzar la máxima calidad posible.

En resumen, aprender a través de vídeos es una forma efectiva y flexible de adquirir conocimientos. Ofrecen una experiencia visual y auditiva que puede ser más atractiva y fácil de entender que un texto, y están actualizados con temas de actualidad. Además, se puede acceder a ellos en cualquier momento y lugar con una conexión a Internet.

CAPÍTULO 3.

METODOLOGÍA

En este capítulo se expondrá y analizará la metodología que se ha seguido para llevar a cabo el presente trabajo y lograr los objetivos propuestos. Una metodología es un conjunto de principios, procesos y técnicas que se utilizan para abordar un problema o realizar una tarea de manera sistemática y efectiva. Se trata de un marco de referencia que guía el pensamiento y la acción de manera que se puedan alcanzar resultados consistentes y predecibles.

La metodología se utiliza a menudo en el ámbito académico y científico para investigar y analizar problemas, pero también se utiliza en otros campos, como la gestión de proyectos, la toma de decisiones y el desarrollo de productos o servicios.

Esta puede incluir una serie de pasos o etapas, como definir el problema, recopilar datos, analizar los datos, formular una hipótesis y llegar a una conclusión. Cada metodología es única y puede variar según el campo o el contexto en el que se utilice. Es importante elegir la metodología adecuada para el problema o la tarea específica que se esté tratando.

3.1. Etapas de la elaboración de un vídeo

A continuación, se presentan algunos pasos que se ha seguido para la elaboración de los vídeos presentados y que ha ayudado a cumplir los objetivos propuestos:

1. **Definir el objetivo del vídeo:** es importante tener un objetivo claro y específico para el vídeo. ¿Qué quieres que aprendan los espectadores? ¿Qué quieres que hagan después de ver el vídeo? Definir el objetivo te ayudará a enfocar el contenido del vídeo y a asegurarte de que está dirigido al público adecuado. Cada vídeo plantea un problema que se resolverá en el mismo y profundizará los conceptos de programación vistos en la propia asignatura.

2. **Seleccionar el nivel o tema del curso en el que se enmarca cada vídeo:** elegir un tema de programación específico y asegurarse de que sea adecuado para el público objetivo del vídeo. Hay que considerar el nivel de conocimiento y experiencia de la audiencia objetivo y elegir un tema que sea apropiado para los mismos. También es importante elegir un tema que sea relevante y de interés para la audiencia. En nuestro caso, cada vídeo se asocia a un tema sobre programación del temario de la propia asignatura.

3. **Investigar y recopilar información:** es imprescindible realizar una investigación exhaustiva sobre el tema para asegurarse de tener una comprensión profunda del mismo. Recopilar materiales relevantes, como tutoriales, libros y artículos en línea, podrán utilizarse como guía durante la creación del vídeo. Hay que citar adecuadamente cualquier fuente de información utilizada y proporcionar enlaces a ellas en la descripción del vídeo.

4. **Escribir un guion:** redactar un guion detallado para el vídeo, incluyendo la estructura, el contenido y el diálogo. Hay que asegurarse de que el guion sea claro y fácil de seguir para la audiencia. Dividir el contenido del vídeo en secciones lógicas y utilizar palabras claves para ayudar a los espectadores a seguir el hilo de la historia. Es aconsejable utilizar un lenguaje sencillo y evitar tecnicismos y palabras complejas si tu audiencia no está familiarizada con ellos. Normalmente, para alcanzar un guion definitivo se ha necesitado de varios borradores previos que han pasado por un proceso de supervisión y corrección.

5. **Grabar el audio:** a la hora de narrar el guion es importante seguir un tono acorde durante toda la grabación, variando el tono de voz y la velocidad para remarcar ciertas palabras claves de la narrativa. Gracias a esto se logra crear un ambiente favorable para seguir el contenido y que no resulte monótono. Es extremadamente importante que el audio siga ciertos requisitos de calidad, tales como:

    - Que tenga un volumen adecuado, especialmente que no sea muy bajo.
    - Que no haya ruido ambiental.
    - Que no se escuchen ruidos puntuales, tales como el teclado, la respiración…
    - Que no haya titubeos al hablar.
    - Que las palabras sean pronunciadas correctamente. Por supuesto, admitiremos como válido el hecho de tener un acento propio de una región hispanohablante, tal y como es el caso del acento andaluz.
    - Que tenga un ritmo adecuado y que las palabras técnicas sean fáciles de entender, especialmente cuando se pronuncian las primeras veces.

6. **Escribir una presentación:** una vez establecido el guion hay que crear una presentación que refleje el contenido narrado del guion. En síntesis, una presentación es lo que el profesor escribiría en la pizarra durante una clase tradicional. La misma de por sí debe ser lo suficientemente clara visualmente para que el espectador sea capaz de comprender el contenido sin necesidad de oír el audio, al menos la parte esencial del mismo. Lo ideal es que el audio sea un complemento que ayude al espectador a afianzar el contenido consumido. Por eso, cuando haya muchas explicaciones es necesario meter texto que vaya escribiendo las mismas. Esto ayuda a reforzar el aprendizaje y facilita al estudiante tanto el acceso a un punto concreto de la explicación, como la acción de parar el vídeo para releer las explicaciones. Es recomendable utilizar conductores visuales que reflejen lo que está narrando el audio, como puede ser señalar con flechas un contenido del vídeo o resaltar el texto en cuestión.

    Destacar que todos los vídeos del canal presentan una estructura similar para la portada y despedida. Para la portada se sigue un diseño parecido al mostrado en la Figura 3.1 donde se presenta una imagen en la que se puede ver el título del tema de la asignatura donde se enmarca el ejercicio o concepto teórico que se va a desarrollar en el propio vídeo. Mientras, se escucha la voz del locutor anunciando lo que se va a explicar a continuación.

Figura 3.1: Portada vídeo: Conjetura de Goldbach. Fuente: Elaboración propia

Para la despedida se sigue el diseño mostrado en la Figura 3.2, donde en la parte superior se muestra el nombre del tema que se ha tratado en el vídeo y en la parte central un texto informativo. Esta diapositiva final se muestra en pantalla durante los últimos 8 segundos del vídeo. Esto se debe a que la plataforma YouTube, en la cual se suben estos vídeos, permite utilizar los últimos segundos para mostrar sugerencias de otros vídeos en pantalla, por lo que dificulta ver el contenido de la presentación. De esta forma, en la mitad inferior de esa diapositiva final aparecerán los dos vídeos sugeridos y no se tapa el texto superior, tal y como se muestra en la Figura 3.3.

7. **Animar la presentación:** el contenido visual debe ser dinámico, pues se trata de un vídeo y no de fotos. Algunas características importantes son:

    - Que sólo se muestre en pantalla la información necesaria para ser consumida, evitando un exceso de información en pantalla que pueda derivar a la pérdida de la narrativa por parte del espectador. Las animaciones más utilizadas se desarrollan con mayor profundidad más adelante.

    - Que el vídeo no esté mucho tiempo sin que se mueva nada. Aunque la voz es importante, ya se ha indicado anteriormente que es esencial reforzarla con imágenes en movimiento.

    - Cuando se pronuncie una palabra técnica, es recomendable que aparezca escrita para reforzar dicha palabra y evitar malentendidos, especialmente las primeras veces que se usen este tipo de vocablos.

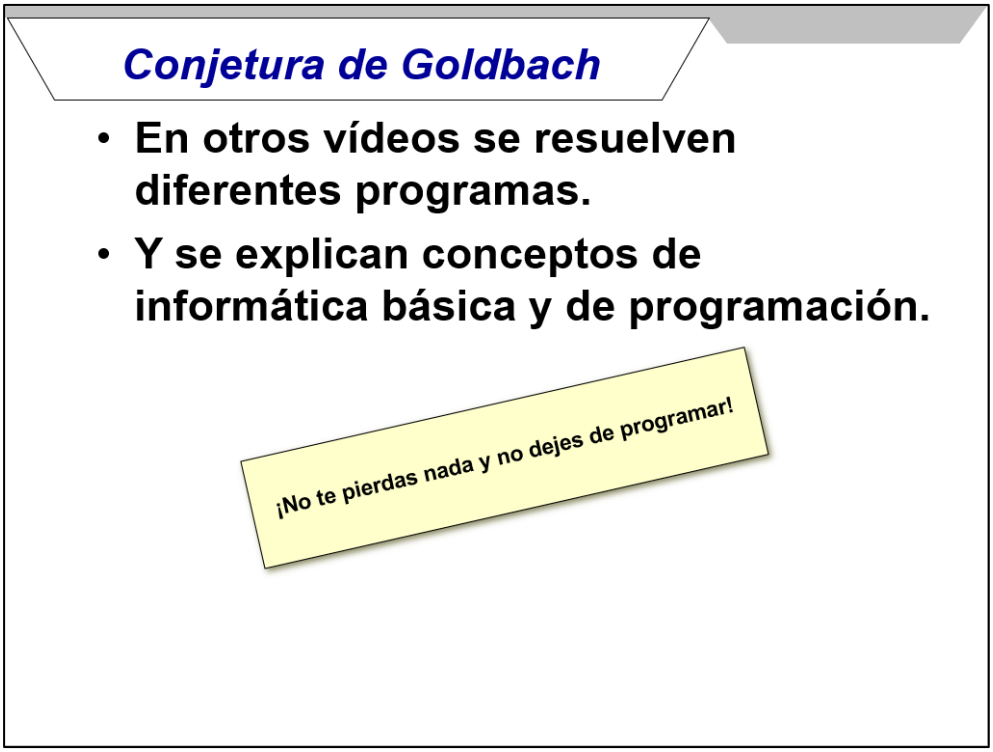

Figura 3.2: Despedida vídeo: Conjetura de Goldbach. Fuente: Elaboración propia

- Cuando las explicaciones hagan referencia a una parte de código, texto u otro objeto que ya esté en pantalla, se debe señalar claramente, para que el estudiante sepa exactamente dónde debe mirar o a qué se refiere la locución.

8. **Cuadrar la presentación con el audio:** este es posiblemente el mayor cuello de botella encontrado durante la elaboración de este trabajo. Una vez creada la presentación con todas las animaciones necesarias, es necesario que encajen con la narrativa del audio. Es decir, hay que cuadrar tanto el tiempo en el que aparecen dichas animaciones en pantalla como la duración de estas. Este proceso se vuelve tedioso y repetitivo debido a la necesidad de probar una y otra vez si la sincronización es correcta. Hay que tener en cuenta las características inherentes del propio software con el que se ha trabajado (PowerPoint), que se analizan más adelante, y detalles como que cuando se retrasa o adelanta la aparición de un objeto, esto puede ocasionar que ese efecto también se aplique a las animaciones posteriores.

9. **Exportar el vídeo:** para exportar el vídeo se ha optado por la máxima calidad permitida dentro del software utilizado (PowerPoint), siendo el caso *"Ultra HD 4K (2880 x 2160)"*, la cual también es compatible con la plataforma objetivo de visionado YouTube. El tiempo de procesamiento para exportar el vídeo con esta configuración es mayor que para el resto de las opciones, pero la calidad alcanzada es notablemente mayor. Hay que tener en cuenta que esta calidad es suficiente para que los textos se vean adecuadamente en el vídeo y que YouTube permite al usuario elegir la calidad del vídeo entre esa y otras inferiores. En otros niveles de calidad, el texto y los gráficos del vídeo pueden verse borrosos.

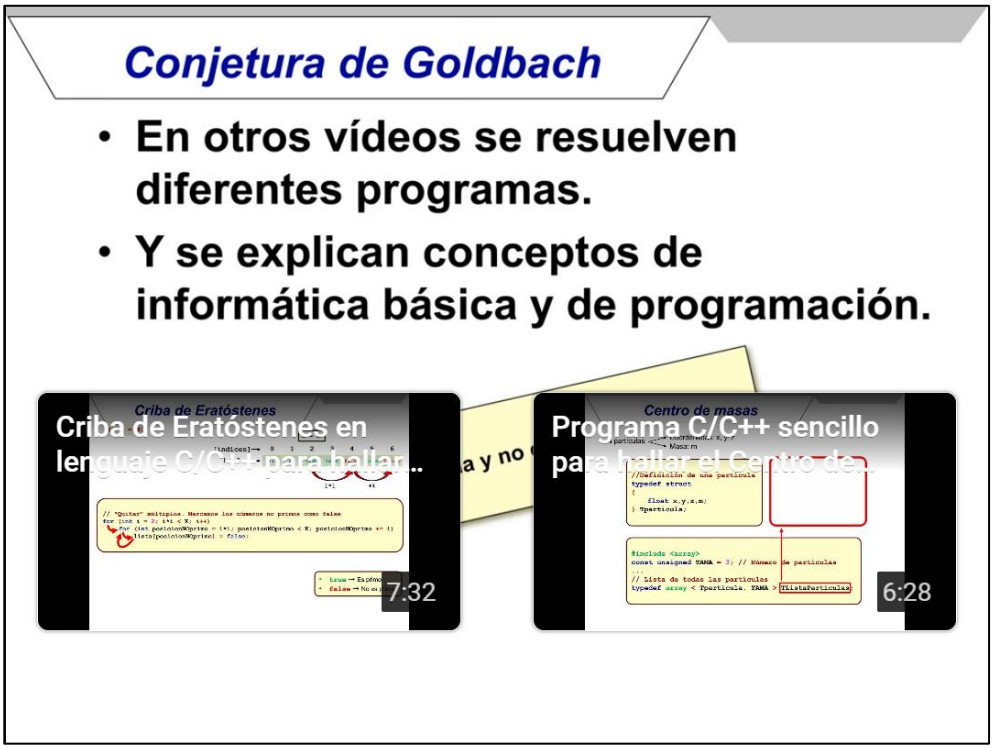

Figura 3.3: Vídeos sugeridos por YouTube en los últimos segundos. Fuente: Elaboración propia

10. **Visionado del vídeo para encontrar errores:** una vez terminado el vídeo es importante visualizar el mismo un par de veces como mínimo para buscar errores. Es posible que durante alguna de las fases del proceso de creación de un vídeo se produzcan ciertos errores que pasen desapercibidos. Es por ello por lo que es esencial visualizar el vídeo prestando gran atención, lo que permite garantizar que el contenido final sea el deseado. En caso de encontrarse algún error, se identifica la fase en la que ha ocurrido y se procede a corregir.

11. **Subir el vídeo a YouTube:** esta fase, de la que se ha encargado el tutor de este trabajo, consiste básicamente en insertar los datos que facilita la plataforma para que el vídeo quede bien identificado. Básicamente, estos datos son:

    - Título del vídeo.
    - Breve resumen de sus contenidos.
    - Lista: indicar si el vídeo pertenece a alguna de las listas existentes en el canal. En nuestro caso hay una lista por cada tema y otra para ejercicios de cada tema.
    - Características generales: idioma, si es un contenido para niños, si tiene contenidos con derechos reservados, etc.
    - Palabras clave del vídeo (para facilitar las búsquedas).
    - Vídeos recomendados al final del vídeo.

- Tarjetas: son vídeos que se sugieren en la parte superior del vídeo, normalmente asociado a alguna parte que se usa en el vídeo actual y que se explica en profundidad en otro vídeo ya existente.

3.2. Investigación y documentación

De acuerdo con la fase 3 expuesta previamente en la metodología, se ha llevado a cabo un proceso exhaustivo de búsqueda de información para adquirir una comprensión profunda de los conceptos de programación necesarios para lograr un contenido de calidad.

Para preparar los vídeos sobre programación en C/C++, se recomienda realizar una investigación exhaustiva sobre el tema y recopilar materiales relevantes que podrán servir como guía durante la creación del vídeo. A continuación, se exponen algunos documentos que se han consultado y podrían ser útiles:

- *Libros de programación:* existen muchos libros o manuales de programación en C y C++ que proporcionan información detallada sobre cómo utilizar estos lenguajes de programación. Este material es especialmente útil porque suelen contener ejercicios acordes al contenido teórico explicado, lo que puede servir de inspiración a la hora de redactar ejercicios propios. Los que hemos empleado en este trabajo son [9, 10, 16]

- *Tutoriales en línea:* hay una gran cantidad de tutoriales en línea sobre programación en C y C++ que pueden ser útiles para aprender sobre estos lenguajes. Muchos de estos tutoriales están diseñados para principiantes y proporcionan ejemplos y explicaciones detalladas de cómo utilizar diferentes características y funcionalidades. Se han consultado los siguientes para entender el funcionamiento y manejo de los software de edición de vídeo utilizados [1,2].

- *Referencias de la biblioteca:* las bibliotecas de C y C++ incluyen una gran cantidad de funciones y características que pueden utilizarse para crear programas. Las referencias de la biblioteca proporcionan información detallada sobre el uso de cada función y su sintaxis.

- *Ejemplos de código:* hay una amplia variedad de ejemplos de código en C y C++ disponibles en Internet que pueden ser útiles para aprender sobre estos lenguajes. Estos ejemplos pueden ser especialmente útiles para entender cómo utilizar diferentes características y funcionalidades de manera práctica [13].

- *Foros y comunidades:* existen muchos foros y comunidades en Internet dedicados a la programación en C y C++ que pueden ser un recurso valioso para aprender sobre estos lenguajes. Estos foros suelen tener una gran cantidad de información y discusiones sobre diferentes aspectos de la programación en C y C++ y pueden ser un lugar útil para obtener ayuda y consejos de otros programadores. Uno de los mayores foros en cuanto a programación se encuentra en la página web “Stack OverFlow” [17].

Además de los materiales presentados anteriormente, se ha utilizado como principal referencia el contenido del temario de la propia asignatura puesto a disposición por el Dpto. de

Lenguajes y Ciencias de la Computación de la Universidad de Málaga [10] a la hora de enfocar y elegir los problemas para la elaboración de los vídeos.

3.3. Herramientas empleadas para la composición de nuestros vídeos

Existen diferentes herramientas, cada cual con sus propias características, que pueden elegirse en función de las necesidades y preferencias tanto del creador del vídeo como del formato del vídeo. Gracias a la documentación previa, se exponen algunas de las herramientas más comunes utilizadas para la creación de vídeos:

- *Software de edición de vídeo:* Los programas de edición de vídeo permiten cortar, unir y añadir efectos a los clips de vídeo. La ventaja que presenta este tipo de software es la escalabilidad que poseen, resulta sencillo utilizar las funciones básicas para alguien que no esté acostumbrado a utilizar este tipo de software, y además, pueden contar con funcionalidades más profundas y complejas que utilizan los profesionales. Algunos ejemplos de software de edición de vídeo son *Adobe Premiere*, *Final Cut Pro* y *Avid Media Composer*. La gran mayoría de software de este tipo suelen ser de pago o disponen de una versión gratuita que genera una marca de agua en el vídeo. Un software interesante y gratuito para esto es *Hit Film.*

- *Software de animación:* Los programas de animación permiten crear vídeos animados y visuales. Estos softwares suelen requerir de conocimiento en animación para poder aprovecharlos al máximo. Algunos ejemplos de software de animación son *Blender, Maya* y *3ds Max.*

- *Software de presentación:* Estos programas permiten crear efectos visuales y añadir elementos como texto y gráficos a los vídeos. Este tipo de software suele ser más fácil e intuitivo que los programas de animación, pero el resultado obtenido puede ser de menor calidad. Algunos ejemplos de software de composición son *PowerPoint* [14], *Adobe After Effects* y *Fusion.*

- *Software de captura de pantalla:* Este tipo de software permite grabar la pantalla del ordenador y el sonido del micro. Resulta especialmente útil utilizar este tipo de software para crear tutoriales y demos. Algunos ejemplos de software de captura de pantalla son *Camtasia* y *OBS Studio*. Es interesante para grabar una explicación sobre un programa concreto y, posteriormente, se puede editar para afinar el producto final (borrar partes del vídeo, añadir efectos, etc.).

- *Hardware de captura de vídeo:* Este tipo de dispositivo permiten grabar el contenido de una cámara o dispositivo externo y transferirlo al ordenador. En nuestro caso, este sería la herramienta que menos se adapta a las necesidades a la hora de crear un vídeo. Algunos ejemplos de hardware de captura de vídeo son *Elgato Game Capture HD60* y *AVerMedia Live Gamer Portable.* Podría ser interesante grabar a la persona que hace cada explicación y mostrarla en ciertos momentos, bien a pantalla completa o en pequeño en una esquina del vídeo. Nosotros hemos optado por no usar esta opción para no perder espacio en pantalla para las explicaciones.

- *Software de grabación de audio:* Estos programas se utilizan para capturar y registrar audio digitalmente. Además, suelen incluir herramientas para editar y mejorar el audio grabado, como la eliminación de ruido de fondo, la corrección de tono y el ajuste del volumen. Algunos ejemplos de estos programas son *Audacity* [3], *Adobe Audition* y *Pro Tools.*

Remarcar que para mantener el formato de los vídeos ya existentes en el canal de YouTube *"Aprende conmigo"* [11], se ha utilizado las mismas herramientas y software para la elaboración de los nuevos vídeos presentados en este trabajo. Esta decisión se toma teniendo en cuenta la necesidad de mantener el formato audiovisual para que no se aprecien cambios significativos en el formato y lograr un estilo homogéneo que se pueda reproducir sin importar quien elabore el vídeo. Estas herramientas se describen en los siguientes apartados.

3.3.1. PowerPoint

PowerPoint es un software de presentación creado por Microsoft [14]. Este software se utiliza principalmente para crear presentaciones de diapositivas que se pueden mostrar en pantalla, proyectar en un proyector o compartir en línea. Estas presentaciones se componen de diapositivas (o páginas) que contienen texto, imágenes, gráficos y otros elementos visuales. Es posible añadir efectos de transición entre las diapositivas y utilizar animaciones individuales para cada elemento con el objetivo de hacer que cada elemento aparezca, desaparezca o se mueva de cierta forma. PowerPoint también permite agregar audio y vídeo en las presentaciones y es compatible con una amplia variedad de formatos de archivo. Es una herramienta muy versátil que se utiliza en diferentes contextos, como puede ser reuniones de negocios, conferencias y aulas de clase.

A la hora de crear los vídeos, la opción que mayor protagonismo ha tenido han sido las animaciones. Son efectos visuales que se añaden a las diapositivas para hacerlas más atractivas y llamar la atención de la audiencia. Las animaciones pueden aplicarse a diferentes elementos de la diapositiva, como texto, imágenes, formas y gráficos. Algunas opciones de animación disponibles en PowerPoint incluyen:

- *Entrada:* Las animaciones de entrada controlan cómo y cuándo aparecen los elementos de la diapositiva. Por ejemplo, se puede configurar una animación de entrada para que una imagen aparezca de forma gradual o para que el texto se desplace hacia la pantalla.

- *Salida:* Las animaciones de salida controlan cómo y cuándo desaparecen los elementos de la diapositiva. Es la función inversa a la animación de entrada. Por ejemplo, se puede configurar una animación de salida para que una imagen desaparezca de forma gradual o para que el texto se desplace hacia fuera de la pantalla.

- *Movimiento:* Las animaciones de movimiento permiten mover elementos de la diapositiva a diferentes posiciones en el momento adecuado. Por ejemplo, se puede configurar una animación de movimiento para que una imagen se desplace a una nueva posición o para que el texto se mueva a lo largo de una trayectoria.

- *Efectos de formato:* Las animaciones de formato permiten cambiar el formato de los elementos de la diapositiva. Por ejemplo, se puede configurar una animación de formato para

que el texto cambie de tamaño o color. También se puede aplicar al resto de elementos descritos.

Para añadir animaciones a una diapositiva de PowerPoint, existe el panel Animaciones del menú Diseño. Es posible configurar diferentes opciones de animación y determinar el orden y el tiempo en el que se reproducen, así como la duración de estas. Esto último cobra especial importancia a la hora de cuadrar las animaciones con el audio, de forma que el contenido visual tenga concordancia con lo que se está escuchando en el audio.

Precisamente, una de las mayores desventajas encontradas a la hora de la elaboración de los vídeos ha sido cuadrar las animaciones y el audio. Esto se debe a que PowerPoint no está diseñado para ser una herramienta de edición de vídeo completa. Resulta fácil y rápido poder crear una presentación, pero cuando en una misma diapositiva hay una gran cantidad de elementos creados, los cuáles poseen animaciones únicas, se vuelve tedioso y complicado poder gestionarlas y cuadrar exactamente el tiempo de dichas animaciones con respecto al audio.

Es por ello, que a lo largo del trabajo, se ha llegado a una optimización en cuanto al problema descrito previamente, solucionándose de la siguiente manera. En vez de tener una única diapositiva con todos los elementos que irán apareciendo y desapareciendo gracias a las animaciones. Se ha optado por dividir en varias diapositivas dichos elementos, de forma que a la hora de gestionarlas el conjunto total de elementos con sus animaciones por diapositiva es menor, permitiendo gestionar con mayor facilidad una animación en concreto y por ende, dedicar menos tiempo a montar y preparar el vídeo.

De igual forma, el audio se ha dividido por el número de diapositivas, es decir, no es recomendable que solo haya una pista de audio para todo el vídeo. Con este planteamiento de trabajo, la pista de audio se ha divido en tantas partes como diapositivas tenga la presentación. Gracias a esto, se ha logrado facilitar enormemente cuadrar las animaciones con el audio. Más aún, una misma diapositiva puede tener varias grabaciones de audio distintas, ya que cada grabación se asocia a un objeto y ese audio sonará cuando aparezca el objeto al que se asocia. Esto facilita la grabación, ya que no es necesario hacer largas locuciones sin errores.

Una vez que la presentación de PowerPoint esté completa, se puede exportar a un formato de vídeo como MP4 o AVI para compartirla en línea o reproducirla en un dispositivo de reproducción de vídeo. Es muy importante tener en cuenta que PowerPoint no es una herramienta de edición de vídeo completa y puede no tener todas las opciones de edición y producción de vídeo disponibles en otros programas.

3.3.2. Audacity

Audacity es un software de edición de audio y grabación gratuito y de código abierto [3]. Es una de las herramientas más populares a la hora de editar archivos de audio, ya que ofrece una amplia variedad de características y sobre todo es muy intuitivo y fácil de usar. Con Audacity, puedes grabar audio, editar archivos de audio existentes, añadir efectos de sonido y mejorar la calidad de una grabación. También puedes utilizarlo para crear archivos de audio para usar en proyectos de sonido y música, como podcasts o composiciones musicales. Este software está disponible para Windows, Mac y Linux.

Durante la elaboración del trabajo se han encontrado varias ventajas al usar Audacity para editar y grabar audio. Dichas ventajas se exponen a continuación:

- **Es gratuito y de código abierto**. Este es un software libre y gratuito, lo que significa que puedes descargarlo y usarlo sin tener que pagar nada.

- **Ofrece una amplia variedad de características**. Tiene muchas opciones de edición y mejora de audio, como cortar, copiar y pegar audio, añadir efectos de sonido y mejorar la calidad del audio.

- **Es fácil de usar**. Tiene una interfaz intuitiva y es fácil de aprender a usar, incluso si no tienes experiencia previa con la edición de audio. Además, en Internet hay una gran variedad de tutoriales para sacar el máximo provecho a este software [8].

- **Es compatible con diferentes plataformas**. Está disponible para Windows, Mac y Linux, por lo que puedes usarlo en la mayoría de los ordenadores.

- **Es útil para una amplia variedad de proyectos**. Es una herramienta versátil y puede ser utilizada para muchos proyectos de audio y música, como podcasts, composiciones musicales y más.

- **No requiere un micro de alta calidad** aunque, por supuesto, ello puede influir en la calidad final.

Aunque, si bien es cierto que Audacity es una herramienta de edición de audio muy popular y potente, también se han encontrado algunas desventajas:

- **Puede ser un poco lento**. A veces, puede ser un poco lento al cargar archivos de audio o al aplicar efectos. Esto puede ser frustrante para algunos usuarios.

- **No es tan completo como algunos programas de pago**. Algunos usuarios pueden sentir que Audacity no tiene características avanzadas que pueden ser interesantes.

- **Puede ser difícil de usar para algunos usuarios**. Aunque es generalmente fácil de usar, algunos usuarios que no estén acostumbrados pueden encontrar su interfaz un poco confusa o complicada.

- **Puede tener problemas de compatibilidad.** A veces, puede tener problemas para abrir archivos de audio en formatos específicos o puede tener problemas de compatibilidad con ciertas versiones de Windows o Mac.

Por otro lado, a la hora de grabar el audio se han utilizados varias opciones de edición y mejora de audio para lograr un resultado más homogéneo y de mayor calidad. De cara a seguir un contenido homogéneo para futuras ampliaciones de vídeos del canal *"Aprende conmigo"*, se describen las características utilizadas a continuación. Nos gustaría destacar que los valores empleados para cada configuración son los mismos que aparecen en las imágenes mostradas:

1. **Reducción de audio**. Por supuesto, es importante hacer las grabaciones en un entorno silencioso. No obstante, a veces puede ser útil reducir el ruido de fondo que pueda contener un archivo de audio. Para ello, se han seguido los siguientes pasos, aunque es importante tener en cuenta que esto puede modificar el resto del sonido de forma indeseada (voz más metálica, por ejemplo):

    1. Selecciona una porción del audio que contenga solo ruido. Es importante que no selecciones ninguna parte del audio que desees conservar.

    2. Haz clic en "Efecto" en la barra de menús y selecciona "Reducción de ruido" de la lista de opciones.

    3. Se abrirá una ventana de diálogo de "Reducción de ruido" como el mostrado en la Figura 3.4 Haz clic en el botón "Obtener perfil de ruido" para que Audacity "identifique" qué sonido es el ruido.

    4. Ajusta los parámetros de reducción de ruido según tus necesidades. Puedes ajustar la cantidad de reducción de ruido, la sensibilidad y la frecuencia de corte.

    5. Haz clic en "Aplicar" para aplicar la reducción de ruido a tu archivo de audio.

    6. Repite los pasos 2 a 6 para cualquier otra porción del audio que desees limpiar.

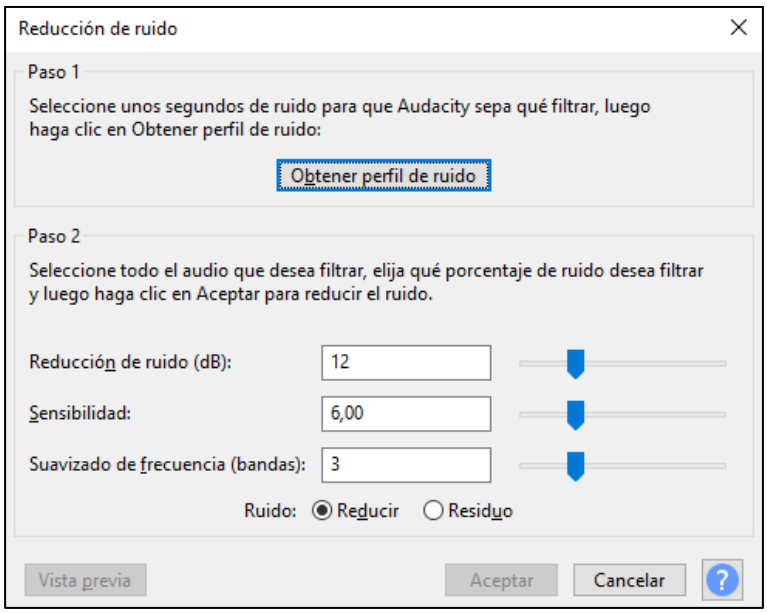

Figura 3.4: Reducción de ruido en Audacity. Fuente: Elaboración propia

2. **Compresor**. Esta configuración te permite ajustar el nivel de sonido de un archivo o pista. Reduce el rango dinámico del audio, lo que significa que reduce la diferencia entre los niveles más altos y bajos del sonido. Esto resulta útil para hacer el sonido más uniforme y fácil de escuchar. Los pasos a seguir son:

1. Selecciona la pista o la porción del audio que deseas comprimir.

2. Haz clic en "Efecto" en la barra de menús y selecciona "Compresor" de la lista de opciones.

3. Se abrirá una ventana de diálogo de "Compresor" como el mostrado en la Figura 3.5 Ajusta los parámetros según tus necesidades. Algunos parámetros comunes para ajustar son el umbral (nivel de sonido a partir del cual el compresor comienza a reducir la señal de audio), la relación (cuánto se reduce la señal de audio por encima del umbral), el ataque (tiempo que tarda el compresor en comenzar a reducir la señal de audio) y la liberación (tiempo que tarda el compresor en dejar de reducir la señal de audio después de que la señal cae por debajo del umbral).

4. Haz clic en "Aplicar" para hacer las modificaciones en tu archivo de audio.

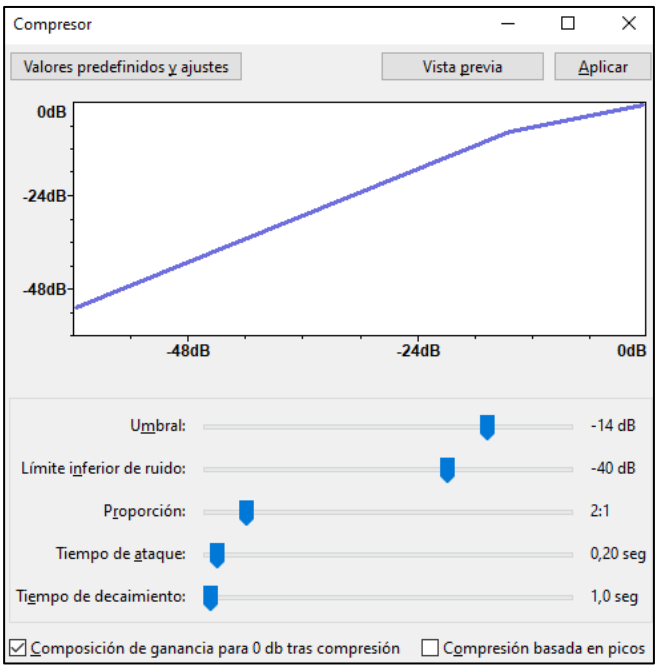

Figura 3.5: Compresor en Audacity. Fuente: Elaboración propia

3. **Ecualizador de curva de filtro**. Esta opción te permite ajustar la intensidad de diferentes bandas de frecuencia en un archivo de audio. Resulta especialmente útil para aumentar o disminuir los niveles de graves o agudos de una grabación. Permitiendo ajustar el tono y la claridad para obtener un resultado más homogéneo y de mejor calidad. Los pasos son:

    1. Selecciona la pista o la porción del audio que deseas ecualizar.

    2. Haz clic en "Efecto" en la barra de menús y selecciona "Ecualizador de curva de filtro" de la lista de opciones.

    3. Se abrirá una ventana de diálogo de "Ecualizador de curva de filtro" como el mostrado en la Figura 3.6.

4. En la gráfica del ecualizador, haga clic en un punto de la curva y arrástrelo para ajustar el nivel de una frecuencia específica.

5. Repita el paso 4 para ajustar el nivel de otras frecuencias según sea necesario.

6. Haga clic en "OK" para aplicar los cambios.

7. Haz clic en "Aplicar" para aplicar el ecualizador de curva de filtro a tu archivo de audio.

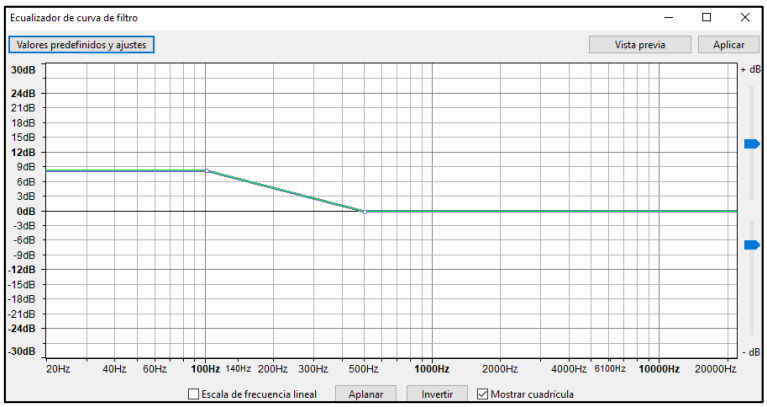

Figura 3.6: Ecualizador de curva de filtro en Audacity. Fuente: Elaboración propia

4. **Limitador**. Esta opción te permite ajustar el nivel máximo de audio de un archivo o pista. Se utiliza para evitar que la señal de audio supere un nivel específico. Resulta útil para evitar distorsión o "clipping"[2] en la grabación, especialmente cuando se está trabajando con niveles de sonido muy altos. Los pasos son:

    1. Selecciona la pista o la porción del audio que deseas limitar.

    2. Haz clic en "Efecto" en la barra de menús y selecciona "Limitador" de la lista de opciones.

    3. Se abrirá una ventana de diálogo de "Limitador" como el mostrado en la Figura 3.7 Ajusta los parámetros según tus necesidades. También se pueden utilizar configuraciones ya predefinidas de limitación.

    4. Haz clic en "Aplicar" para aplicar el limitador a tu archivo de audio.

[2] Wikipedia. (s. f.). Clipping (audio) - Wikipedia. Recuperado 6 de enero de 2023, de https://en.wikipedia.org/wiki/Clipping_(audio)

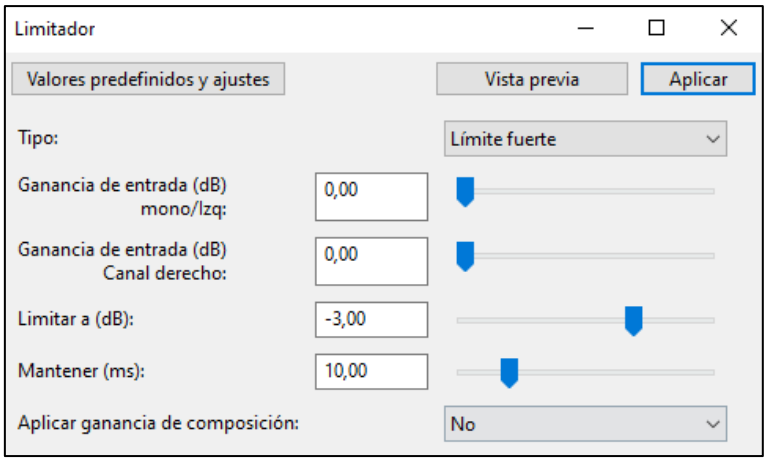

Figura 3.7: Limitador en Audacity. Fuente: Elaboración propia

5. **Normalizar**. Esta opción te permite ajustar el nivel de sonido de una grabación para que tenga un nivel máximo específico. Se utiliza para asegurar que todas las grabaciones tengan el mismo nivel de volumen. lo que puede ser útil cuando se están combinando varias pistas o para hacer que la grabación sea más fácil de escuchar.

   Destacar la diferencia entre, la normalización, que se utiliza para ajustar el volumen de una grabación y el limitador, que se utiliza para evitar distorsión y controlar el nivel de sonido en una grabación.

   Los pasos para aplicar este ajuste son:

   1. Selecciona la pista o la porción del audio que deseas normalizar.

   2. Haz clic en "Efecto" en la barra de menús y selecciona "Normalizar" de la lista de opciones.

   3. Se abrirá una ventana de diálogo de "Normalizar" como el mostrado en la Figura 3.8 Ajusta el nivel máximo de volumen deseado en la opción "Nivel máximo". Puedes dejar la opción "Eliminar silencios" marcada si deseas eliminar cualquier silencio al principio o al final del archivo de audio.

   4. Haz clic en "Aplicar" para aplicar la normalización a tu archivo de audio.

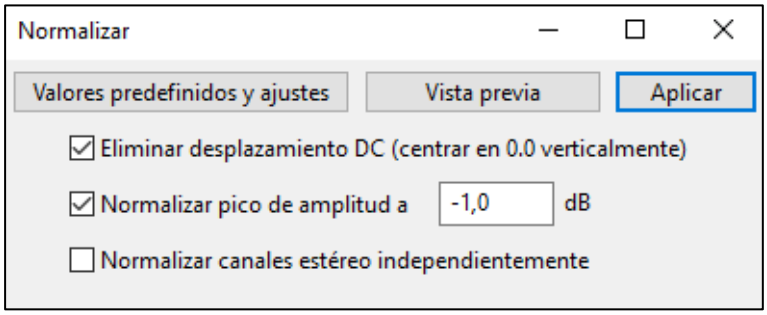

Figura 3.8: Normalizar en Audacity. Fuente: Elaboración propia

Es importante tener en cuenta que todas estas configuraciones pueden afectar a la calidad del audio si se utiliza en exceso, por lo que es importante comprobar cuidadosamente el audio después de aplicar los efectos para asegurarse de que el resultado es el deseado.

En la Figura 3.9 se muestra la diferencia de una pista de audio antes de editar (pista de audio superior) y después de aplicar las mejoras comentadas (pista de audio inferior). A primera vista se puede observar una diferencia notable en la amplitud de señal de ambas pistas.

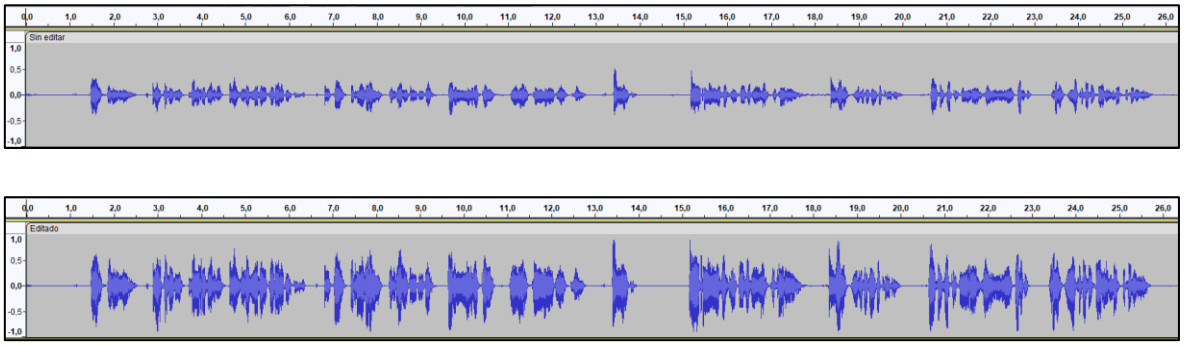

Figura 3.9: Amplitud de señal de una grabación antes y después de ser editada. Fuente: Elaboración propia

3.3.3. Code::Blocks

Code::Blocks es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) de código abierto para programar en diferentes lenguajes de programación, como C, C++, Fortran y otros [7]. Es compatible con Windows, Linux y MacOS y ofrece una amplia gama de características para facilitar el desarrollo de aplicaciones, como la edición de código, el depurado, la compilación de programas y herramientas de construcción para facilitar el proceso de desarrollo de aplicaciones.

Este entorno de desarrollo se utiliza para impartir las clases de programación en la Escuela de Ingenierías Industriales de la Universidad de Málaga y, como tal, también se ha utilizado para la elaboración de este trabajo. Este entorno permite ejecutar, depurar y compilar todos los códigos mostrados a lo largo de los vídeos presentados.

Para utilizar Code::Blocks, primero necesitas descargar e instalar el software en tu ordenador. Una vez instalado, puedes crear un nuevo proyecto, seleccionando el tipo de proyecto que deseas crear y siguiendo las instrucciones en pantalla para configurarlo. Una vez realizado este proceso podrás comenzar a escribir y editar código. Si el programa solo tiene un fichero, no es necesario crear un proyecto, pudiendo crear un fichero individual que, en nuestro caso, tendrá la extensión .CPP. Este entorno de desarrollo ofrece un editor de código con resaltado de sintaxis y autocompletado de código, así como un depurador para ayudarte a encontrar y solucionar errores en tu programa.

Además, Code::Blocks incluye un compilador integrado para compilar y ejecutar tu código. Puedes configurar diferentes opciones de compilación para personalizar el proceso de compilación de acuerdo con tus necesidades. Cuando hayas terminado de escribir tu código, puedes compilar y ejecutar tu aplicación haciendo clic en el botón "Compilar y ejecutar" en la barra de herramientas o utilizando el atajo de teclado correspondiente. Si encuentras errores en tu código, puedes utilizar el depurador de Code::Blocks para ayudarte a encontrar y solucionar los problemas.

Por su simplicidad y facilidad de uso, Code::Blocks es una herramienta muy útil para programadores de C y C++, y es muy popular entre los desarrolladores que utilizan estos lenguajes. Si estás interesado en aprender a programar en C o C++, Code::Blocks puede ser una excelente opción para comenzar, además su principal ventaja es que es de código abierto, por lo que podrás disponer de este software totalmente gratis.

CAPÍTULO 4.

MATERIAL AUDIOVISUAL CREADO

En este capítulo se presentan los vídeos creados para el presente TFG, así como un marco previo de los vídeos ya existentes en el canal de YouTube "*Aprende conmigo*" [11].

Debido a la complejidad de trasladar un contenido audiovisual a un formato solamente escrito, se ha optado por seguir la siguiente estructura.

A modo de contexto, se expondrá un breve resumen de los temas de la asignatura Fundamentos de Informática, los cuales han servido de marco teórico para enfocar el contenido de los vídeos. Posteriormente se enumerarán los vídeos creados para cada tema, dónde se mostrará un breve resumen del contenido del vídeo, así como una imagen significativa y el código empleado en el mismo La lista de vídeos con sus direcciones URL específicas se muestran en el Anexo 1.

4.1. Material audiovisual ya existente en el canal "*Aprende conmigo*"

El tutor de este TFG viene desarrollando contenido audiovisual sobre programación en C/C++, abarcando principalmente el contenido teórico de la asignatura Fundamentos de Informática que se ofrece a los alumnos a lo largo del curso. Este contenido multimedia esta subido en el canal de YouTube "*Aprende conmigo*" [11] y resulta un recurso útil para aquellos interesados en aprender programación en lenguaje C/C++. Los vídeos del canal ofrecen explicaciones detalladas de diferentes conceptos y también presentan soluciones a ejercicios relacionados.

A fecha de entrega del presente TFG, el canal cuenta con **45 vídeos** subidos, de los cuales 14 han sido creados por el autor de este trabajo. Los vídeos que se encontraban disponibles en el canal antes de añadir los nuevos 14, se centraban casi por completo en ofrecer explicaciones teóricas sobre los diferentes temas de la asignatura de Fundamentos de Informática, profundizando y ayudando al espectador a comprender dichos conceptos a través de explicaciones profundas y ejemplos prácticos.

Para facilitar la búsqueda de contenido en el canal, se han creado listas de reproducción. Estas son una forma de agrupar vídeos relacionados en una sola colección. Las listas resultan especialmente útiles para organizar los vídeos del canal de YouTube en categorías temáticas o para crear una secuencia de vídeos que deben ver los espectadores en un orden específico. En nuestro caso hay una lista por cada tema y otra para ejercicios de cada tema. En la Tabla 4.1 se muestran las listas de reproducción creadas antes del presente TFG con el número de vídeos presentes en las mismas.

Listas de reproducción	Número de vídeos
Tema 1. El ordenador y la información: hardware y software	7
Tema 2. Algoritmos y programas (introducción a la programación)	6
Tema 4. Estructuras de control en Lenguaje C/C++	10
Tema 5. Funciones en Lenguaje C/C++	3
Tema 6. Arrays y estructuras (Lenguaje C/C++)	5

Tabla 4.1: Listas de reproducción del canal "Aprende conmigo" antes del presente TFG. Nota: observe que no había vídeos del Tema 3. Fuente: [11]

A modo de comparativa, se expone en la Tabla 4.2 el número de vídeos total tras la realización de este TFG.

Lista de reproducción	Número de vídeos
Tema 1: El ordenador y la información: hardware y software	7
Tema 2: Algoritmos y programas (introducción a la programación)	6
Tema 3: Introducción al Lenguaje C/C++	**2**
Tema 4: Estructuras de control en Lenguaje C/C++	10
Tema 5: Funciones en Lenguaje C/C++	3
Tema 6: Arrays y estructuras (Lenguaje C/C++)	5
Ejercicios del Tema 3: Programas básicos en C/C++	**4**
Ejercicios del Tema 4: Estructuras de selección y de bucle en programas simples en C/C++	**4**
Ejercicios del Tema 5: Funciones en Lenguaje C/C++	**2**
Ejercicios del Tema 6: Arrays y estructuras en Lenguaje C/C++	**2**

Tabla 4.2: Listas de reproducción del canal "Aprende conmigo" después del presente TFG. Se marcan en negrita los temas que se han incorporado. Fuente: [11]

4.2. Tema 3. Introducción a C/C++

En este tema se introduce el lenguaje de programación C/C++, el cual es un lenguaje de programación de alto nivel y de propósito general que se originó en los años 70. C++ es un lenguaje de programación orientado a objetos que fue desarrollado a partir de C en 1979. Ambos son lenguajes muy populares y se utilizan ampliamente en aplicaciones de sistemas y aplicaciones de alto rendimiento, como juegos y aplicaciones empresariales.

Una de las principales características de C y C++ es que son lenguajes de programación compilados, lo que significa que el código fuente escrito por el programador se compila o traduce en código máquina para que pueda ser ejecutado por una computadora. Esto permite que el código sea más rápido y eficiente, pero también significa que hay que pasar por la fase de compilación y que el código objeto resultante es difícil de leer y entender para las personas (aunque no es necesario).

Otra característica importante de C y C++ es que son lenguajes de programación de alto nivel, pero tienen características de “bajo nivel”, lo que significa que para ciertas operaciones están más cerca del lenguaje de la máquina y tienen un control más fino sobre el hardware de la computadora. Esto los hace adecuados para aplicaciones que requieren un alto rendimiento o que deben interactuar directamente con el hardware. Sin embargo, también significa que son más difíciles de usar y pueden ser más propensos a errores y fallos.

Aunque la mayoría de lo que se enseña en este curso es propio de C estándar, ciertas operaciones se hacen en C++ (por ejemplo, las operaciones de lectura y escritura con cin y cout). Por eso, no podemos hablar de que nuestros programas son en C estándar, sino que tenemos que decir que son programas en C++, o bien, en C/C++. Así pues, la extensión de nuestros ficheros de programación será .CPP (de C++, y no simplemente .C).

En este tema también se introduce por primera vez los conceptos básicos de variables y operadores. Una variable es un espacio de almacenamiento en la memoria de la computadora donde se puede guardar un valor. Las variables deben ser declaradas antes de poder ser utilizadas en un programa y deben tener un tipo de datos específico, que indica qué tipo de valor pueden almacenar.

Algunos ejemplos de declaración de variables en C++ pueden verse en la Tabla 4.3.

Declaración de variables	Comentario
`int a;`	Declara una variable entere llamada "a"
`float b;`	Declara una variable de tipo punto flotante llamada "b"
`char c;`	Declara una variable de tipo carácter llamada "c"
`string s;`	Declara una variable de cadena de caracteres llamada "s"

Tabla 4.3: Declaración de variables en C++. Fuente: Elaboración propia

Una vez que se ha declarado una variable, se puede asignarle un valor usando el operador de asignación "=". En la Tabla 4.4 se pueden ver algunos ejemplos.

Asignar un valor a una variable	Comentario
`a = 5;`	Asigna el valor 5 a la variable "a"
`b = 3.14;`	Asigna el valor 3.14 a la variable "b"
`c = 'x';`	Asigna el carácter 'x' a la variable "c"
`s = "Hola";`	Asigna la cadena "Hola" a la variable "s"

Tabla 4.4: Asignación de un valor a una variable en C++. Fuente: Elaboración propia

Además de la asignación, hay muchos otros operadores que se pueden utilizar en C++ para realizar cálculos y manipular variables. Algunos ejemplos comunes se muestran en la Tabla 4.5.

Operadores	Carácter reservado
Aritméticos	`+ (suma), - (resta), * (multiplicación), / (división), % (módulo)`
Asignación	`+=, -=, *=, /=, %=`
Comparación	`== (igual), != (distinto), > (mayor que), < (menor que), >= (mayor o igual que), <= (menor o igual que)`
Lógicos	`&& (y), || (o), ! (no)`

Tabla 4.5: Ejemplo de algunos operadores en C++. Fuente: Elaboración propia

Es importante recordar que cada tipo de datos tiene sus propias reglas y restricciones en cuanto a qué operaciones se pueden realizar y qué tipo de resultados se obtienen.

4.2.1. Vídeo: Operaciones básicas del tipo string en C/C++

En este vídeo se explica lo más básico del tipo string (cadenas de caracteres) y algunas de las operaciones más usuales:

- Declaración de variables string y asignación de valores.
- Constantes de tipo string (con comillas dobles). Recuerde que las comillas simples se usan para las constantes de tipo char.
- Concatenación de cadenas (operador +).
- Longitud de una cadena: función length().
- Acceder y modificar un carácter con los corchetes: Variable[posición].
- Extraer una subcadena: función substr().
- Escritura de cadenas con cout.
- Lectura con cin y con la función getline().

En la Figura 4.1 se puede apreciar una imagen estática de este vídeo en la cual se están explicando conceptos relacionados con el tipo de dato string.

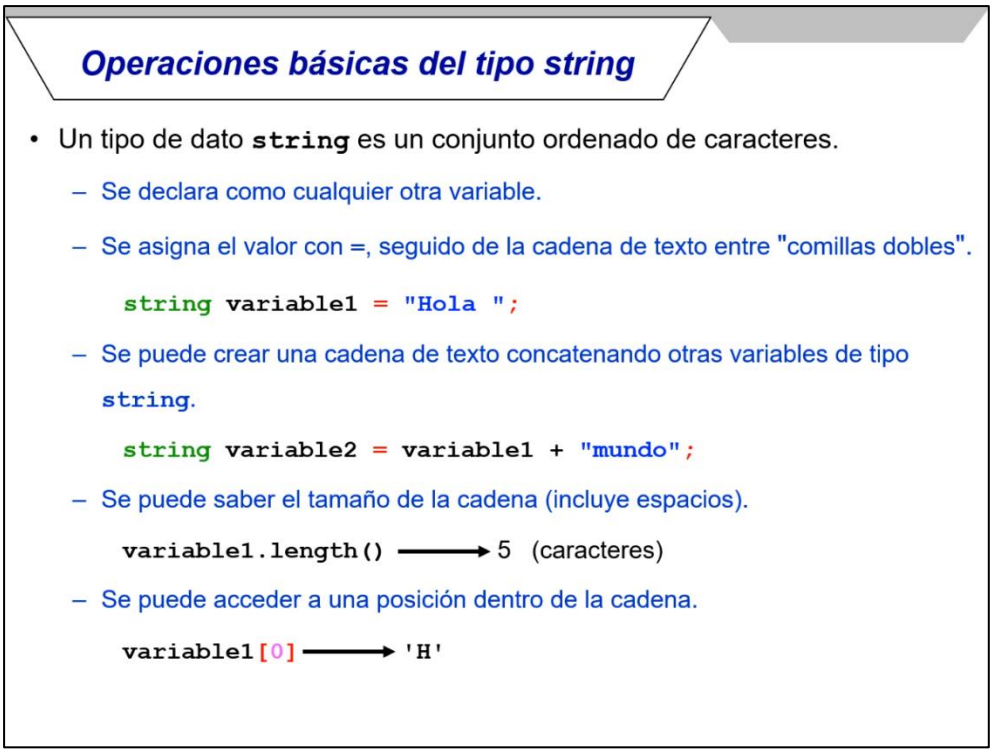

Figura 4.1: Una imagen de ejemplo del vídeo "Operaciones básicas del tipo string". Fuente: Anexo 1.

- **Código:**

```
// Programa que ejecuta operaciones básicas del tipo string

#include <iostream>
#include <string>
using namespace std;

int main ()
{
    string variable1, variable2;
    variable1 = "Hola ";
    variable2 = variable1 + "mundo";
    cout<<endl<<endl<<endl;
    cout << "  Al concatenar obtenemos: " << variable2 << endl;
    cout << "  Longitud: " << variable1.length() << endl;
    cout << "  Subcadena: " << variable1.substr(2,2) << endl;
    cout << "  Accedemos al primer carácter: " << variable1[0]
    << endl;
    variable1[0] = 'A';
    cout << "  Modificamos el primer carácter: " << variable1
    << endl<<endl<<endl;
    return 0;
}
```

4.2.2. Vídeo: Operadores ++ y -- en C/C++ (preincremento, preincremento, postincremento, predecremento y postdecremento)

En este vídeo se explican los operadores de incremento `(++)` y decremento `(--)`, en sus dos versiones cada uno (pre y post). Además, se incluyen:

- Ejemplos.
- Explicación de la precedencia y de la asociatividad de operadores.

En la Figura 4.2 se puede apreciar una imagen estática de este vídeo en la cual se explican conceptos relacionados con los operadores de incremento y decremento.

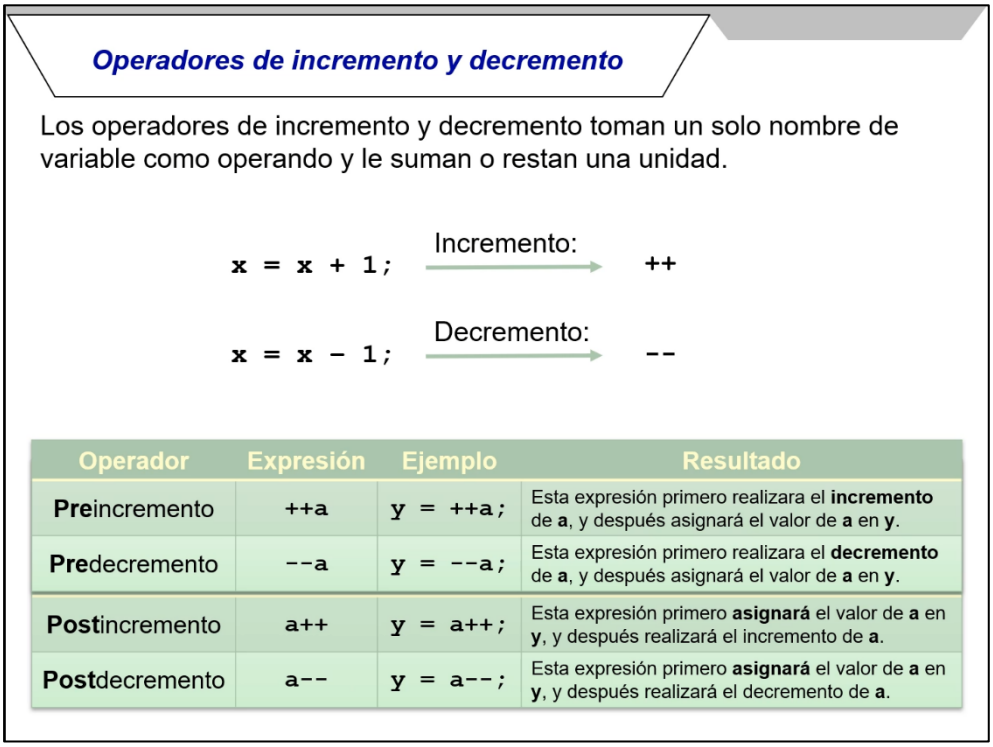

Figura 4.2: Una imagen de ejemplo del vídeo "Operadores de incremento y decremento". Fuente: Anexo 1.

- **Código:**

```
// Programa que ejecuta distintas operaciones con operadores
de incremento y decremento

#include <iostream>
using namespace std;


int main ()
```

```
{
    int a=7, b;
    b = a++;
    cout <<endl<<endl<<endl<<endl<< "     El valor de b es: "
    << b << " y el de a es: " << a <<endl;
    b = ++a;
    cout << "     El valor de b es: " << b << " y el de a es:
    " << a <<endl;
    cout <<endl<<endl<<endl <<"     Expresión con incremento
    y decremento: " << ++a - b-- << endl;
    return 0;
}
```

4.2.3. Vídeo: Operaciones simples entre enteros y reales (tipos de datos int, float, double y long double)

En este vídeo se explican los tipos de datos int, float, double y long double, y se muestran ejemplos de operaciones sobre estos tipos. En particular, se explica:

- Tipos de datos, rangos y la notación científica (o exponencial).
- División entre enteros (sin sacar decimales: división entera).
- Resto de la división entera.
- División y multiplicación entre un entero y un real (saca decimales).
- Asignación de un número real a una variable entera (se pierden los decimales).
- Utilización de un molde (casting o moldeado) para cambiar el tipo de una expresión.

En la Figura 4.3 se puede apreciar una imagen estática de este vídeo en la cual se explican conceptos relacionados con los tipos de datos numéricos.

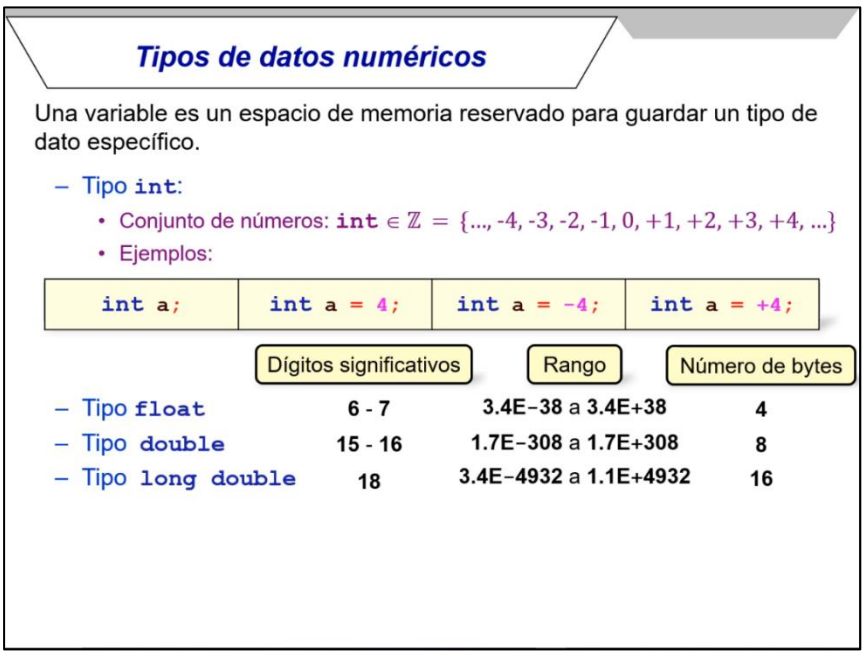

Figura 4.3: Una imagen de ejemplo del vídeo: "Operaciones simples entre enteros y reales". Fuente: Anexo 1.

- **Código:**

```
// Operaciones simples entre distintos tipos de datos

#include <iostream>
using namespace std;
int main ()
{
     int a=3, b=6, c, d, e;
     float f=2.2, g, h;
     c = a / b;
     d = a % b;
     e = a * f;
     g = a / b;
     h = a / float(b);
     cout << "El resultado de c es: " << c << endl; // 0
     cout << "El resultado de d es: " << d << endl; // 3
     cout << "El resultado de e es: " << e << endl; // 6
     cout << "El resultado de g es: " << g << endl; // 0
     cout << "El resultado de h es: " << h << endl; // 0.5
     return 0;
}
```

4.2.4. Vídeo: Programa C/C++ simple sobre el tipo string

En este vídeo se explica cómo podemos leer 3 palabras (3 cadenas de caracteres de tipo string), y cómo calcular el número de caracteres total. Usamos el operador + que aplicado al tipo string lo que hace es concatenar las cadenas (pone la cadena de la derecha al final de la cadena de la izquierda). En la Figura 4.4 se puede apreciar una imagen estática de este vídeo en la cual se observa tanto el enunciado del problema como su solución.

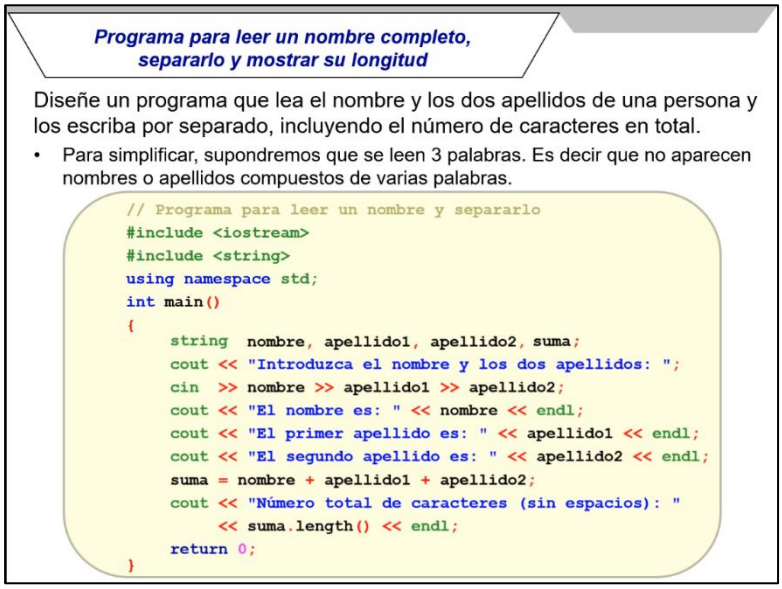

Figura 4.4. Una imagen de ejemplo del vídeo: "Programa C/C++ simple sobre el tipo de dato string". Fuente: Anexo 1.

- **Código:**

```
// Programa para leer un nombre completo, separarlo y mostrar
su longitud
// Observaciones: Solo funciona para nombres simples con
apellidos simples (de una palabra cada uno)
#include <iostream>
using namespace std;

int main()
{
    string a, b, c, d;
    cout << "Introduzca el nombre y los dos apellidos: ";
    cin  >> a >> b >> c;
    cout << "El nombre es: " << a << endl;
    cout << "El primer apellido es: " << b << endl;
    cout << "El segundo apellido es: " << c << endl;
    d = a + b + c;
    cout << "Número total de caracteres (sin espacios): " <<
    d.length();
    return 0;
}
```

4.2.5. Vídeo: Programa C/C++ para ver el código de un carácter (molde int) y su tamaño en bytes (operador sizeof)

En este vídeo se explica cómo podemos usar un molde (casting) para acceder al código entero de un carácter almacenado en una variable de tipo char. También se muestra cómo descubrir el tamaño en byte de cualquier variable (o expresión), usando el operador sizeof. El vídeo muestra la tabla de caracteres ASCII de 7 y de 8 bits.

En la Figura 4.5 se puede apreciar una imagen estática de este vídeo en la cual se muestra una explicación acerca de la tabla de caracteres ASCII de 7 bits.

- **Código:**

```
// Programa que lee un carácter y devuelve su código ASCII y
el tamaño en bytes

#include <iostream>
using namespace std;

int main ()
{
    char a;
    cout << "Introduce cualquier carácter: ";
    cin >> a;
    cout << "- El código de ese carácter es: " << int(a) <<
    endl;
    cout << "- Bytes requeridos: " << sizeof(a) << endl;
    return 0;
}
```

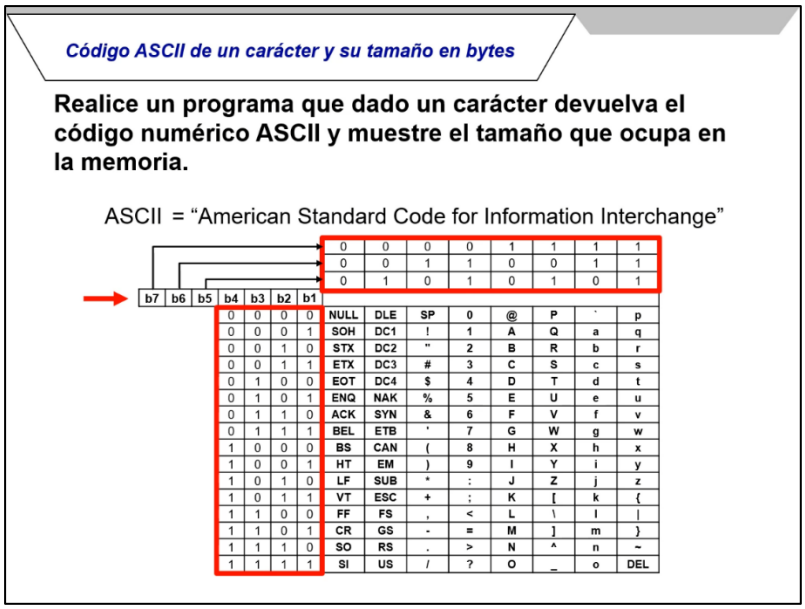

Figura 4.5 Una imagen de ejemplo del vídeo: "Programa C/C++ para ver el código de un carácter y su tamaño en bytes". Fuente: Anexo 1.

4.2.6. Vídeo: Programa C/C++ para resolver una ecuación de segundo grado

En este vídeo se explica cómo calcular las dos soluciones reales de una ecuación de segundo grado a partir de sus tres coeficientes: ax^2 + bx + c = 0. Para mejorar este ejercicio, se usa la sentencia if, la cual se explica brevemente. En otro vídeo se explica la sentencia if con más detalle.

En la Figura 4.6 se puede apreciar una imagen estática de este vídeo en la cual se explica cómo resolver una ecuación de segundo grado en lenguaje C/C++.

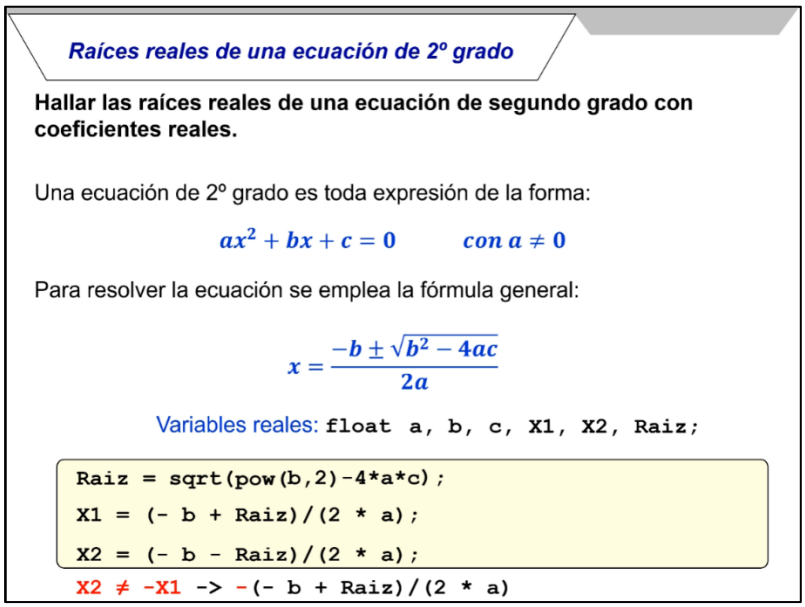

Figura 4.6: Una imagen de ejemplo del vídeo: "Programa C/C++ para resolver una ecuación de segundo grado". Fuente: Anexo 1.

- **Código:**

```
// Programa que resuelve una ecuación de segundo grado

#include <iostream>
#include <cmath>

using namespace std;

int main()
{
   cout<<variable;
   int a, b, c, X1 ,X2 ,Raiz;
   cout << "Introduzca la variable a:";
   cin >> a;
   cout << "Introduzca la variable b:";
   cin >> b;
   cout << "Introduzca la variable c:";
   cin >> c;
   if ((a == 0) ||((pow(b,2)-4*a*c)<0)){
      cout << "No se puede resolver";
   } else {
       Raiz = sqrt(pow(b,2)-4*a*c);
       X1 = (-b+Raiz)/(2*a);
       X2 = (-b-Raiz)/(2*a);
       cout << "Las soluciones son: " << X1 << " y " << X2 << endl;
   }
   return 0;
}
```

4.3. Tema 4. Estructuras de Control

En este tema se explican las diferentes estructuras de control, como son las estructuras de selección (condicionales), y las repetitivas o iterativas (bucles). También se explica el anidamiento de bucles.

Las estructuras de selección en C++ permiten a un programa tomar decisiones y ejecutar ciertas acciones en función de una condición dada. Las estructuras de selección son el if y el switch.

La estructura if se usa para evaluar una condición y ejecutar una o más declaraciones si la condición se evalúa como verdadera. La sintaxis es la siguiente:

```
if (condición) {
  // código a ejecutar si la condición es verdadera
}
```

También puede usar la estructura if-else para especificar qué código ejecutar si la condición es verdadera o falsa:

```
if (condición) {
  // código a ejecutar si la condición es verdadera
} else {
  // código a ejecutar si la condición es falsa
}
```

La estructura "switch" es similar a if-else pero se usa cuando hay varias condiciones posibles y se quiere ejecutar una acción diferente para cada una de ellas. La sintaxis es la siguiente:

```
switch (expresión) {
  case valor1:
    // código a ejecutar si la expresión es igual a valor1
    break;
  case valor2:
    // código a ejecutar si la expresión es igual a valor2
    break;
  ...
  default:
    // código a ejecutar si ninguno de los casos anteriores
    // se cumple
    break;
}
```

Las estructuras repetitivas o iterativas, también conocidas como bucles, permiten ejecutar un bloque de código varias veces. Las estructuras repetitivas más comunes en C++ son "while", "do-while" y "for". La estructura "while" ejecuta un bloque de código mientras se cumpla una condición dada. La sintaxis es la siguiente:

```
while (condición) {
  // código a ejecutar mientras se cumpla la condición
}
```

La estructura do-while es similar a while, pero asegura que el bloque de código se ejecute al menos una vez, ya que primero ejecuta el código y luego evalúa la condición. La sintaxis es la siguiente:

```
do {
  // código a ejecutar al menos una vez
  // ... y mientras se cumpla la condición
} while (condición)
```

4.3.1. Vídeo: Programa C/C++ para contar cuántas veces aparece un carácter en una frase

En este vídeo se explica un programa sencillo que lee una frase (en una variable string) y un carácter (variable tipo char) y cuenta cuántas veces aparece ese carácter en la frase:

- Se explican las formas de leer un carácter y una sucesión de caracteres (tipo string).

- Se explica brevemente cómo funciona el bucle for (el cual se explica con detenimiento en otro vídeo).
- Se explica la función length() para calcular el número de caracteres de una variable tipo string.

En la Figura 4.7 se puede apreciar una imagen estática de este vídeo en la cual se explica cómo poder almacenar y acceder a un carácter y una frase en lenguaje C/C++.

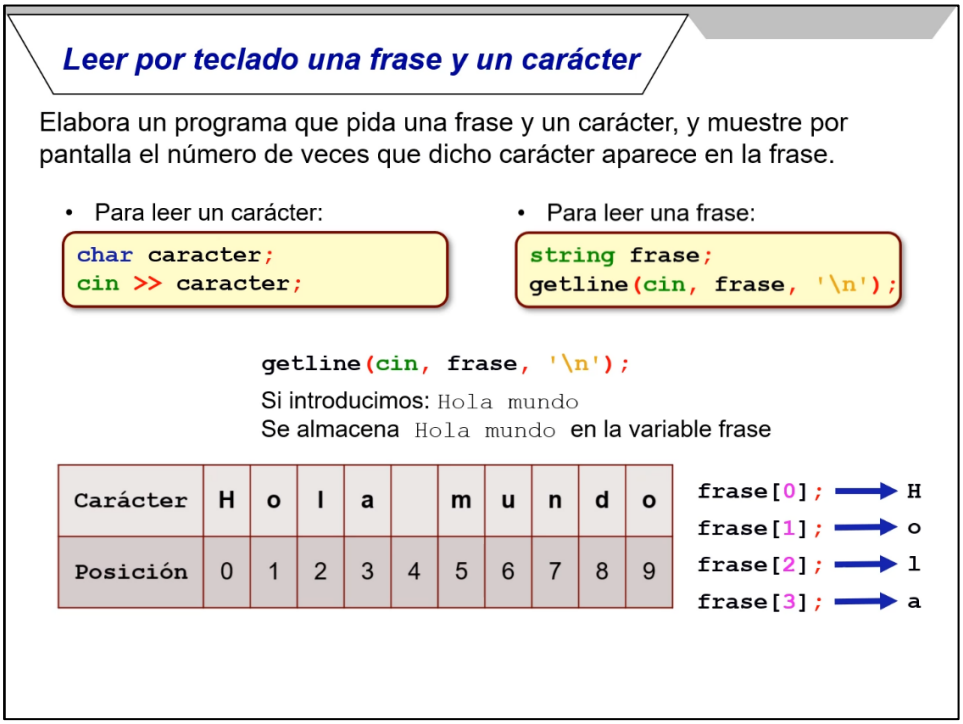

Figura 4.7: Una imagen de ejemplo del vídeo: "Programa C/C++ para contar cuántas veces aparece un carácter en una frase". Fuente: Anexo 1.

- **Código:**

```
//Programa que cuenta las veces que aparece un carácter en una
cadena de texto.

#include <iostream>
#include <string>
using namespace std;

int main()
{
    string frase;
    char caracter;
    int contador = 0;
    cout << "Introduzca una frase: ";
    getline(cin, frase, '\n');
    cout << "Introduzca el carácter que quiera contar: ";
    cin >> caracter;
```

```
        for(int i = 0; i < frase.length(); i++)
            if(caracter == frase[i])
                contador++;
        if (contador == 1)
             cout << "La letra " << caracter << " apareció " <<
             contador << " vez.";
        else
             cout << "La letra " << caracter << " apareció " <<
             contador << " veces.";
        return 0;
    }
```

4.3.2. Vídeo: Programa C/C++ para calcular las raíces de una función matemática por el método de Newton-Raphson

En este vídeo se explica un programa sencillo que calcula el valor que hace cero una función matemática usando el método de Newton-Raphson:

- Se trata de un método iterativo que se va acercando a la solución en cada ejecución.
- Se usa y se explica un bucle while que termina cuando nos acercamos suficientemente a la solución o cuando demos un máximo de 30 iteraciones. El bucle while se explica con detenimiento en otro vídeo.
- Se explican los operadores `&&` (AND) y `||` (OR) de lenguaje C.
- Se recalca la importancia de modificar las variables de la condición de un bucle, para garantizar que el bucle termina, evitando que sea un bucle infinito (un bucle que no termina, porque su condición siempre es verdad).

En la Figura 4. se puede apreciar una imagen estática de este vídeo en la cual se explica y muestra el código completo que resuelve el problema planteado.

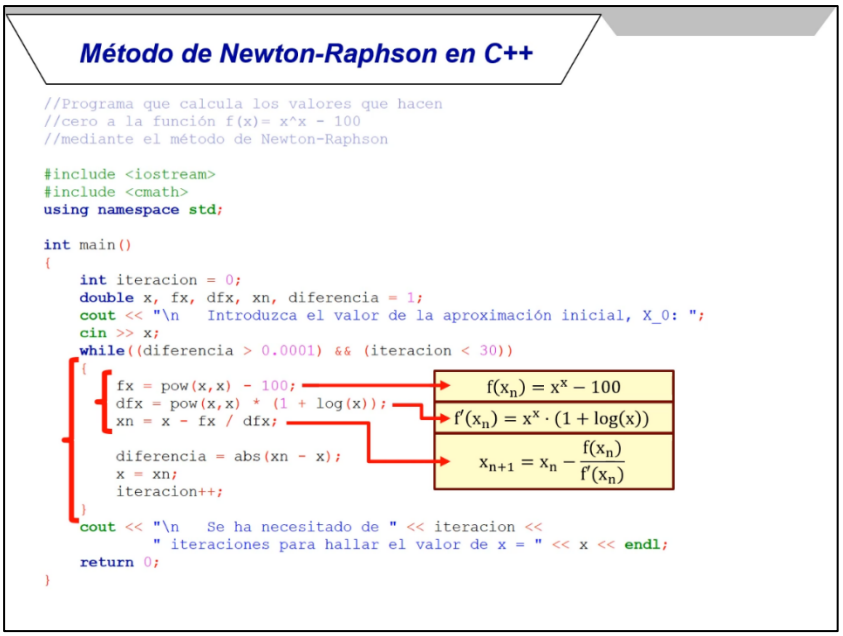

Figura 4.8: Una imagen de ejemplo del vídeo: "Programa C/C++ para calcular las raíces de una función matemática por el método de Newton-Raphson". Fuente: Anexo 1.

- **Código:**

```
//Programa que resuelve la función f(x)= x^x - 100 Mediante el
método de Newton-Raphson

#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    int iteracion = 0;
    double x, fx, dfx, xn, diferencia = 1;
    cout << "\n   Introduzca el valor de la aproximación
    inicial, X_0: ";
    cin >> x;
    while((diferencia > 0.0001) && (iteracion < 30))
    {
        fx = pow(x,x) - 100;
        dfx = pow(x,x) * (1 + log(x));
        xn = x - fx / dfx;

        diferencia = abs(xn - x);
        x = xn;
        iteracion++;
    }
    cout << "\n   Se ha necesitado de " << iteracion <<
            " iteraciones para hallar el valor de x = " <<
            X << endl << endl;
    return 0;
}
```

4.3.3. Vídeo: Dibujar un triángulo de asteriscos (relleno y hueco)

En este vídeo se explican dos programas sencillos que dibujan un triángulo con asteriscos. El primer triángulo está relleno de asteriscos completamente, mientras que el segundo escribe solo los bordes. Además, se muestra:

- Cómo manejar bucles anidados, uno dentro de otro y usar la variable de control del bucle principal en la condición de los bucles interiores. Los bucles anidados se explican en detalle en otro de los vídeos.
- Se introduce una sentencia if dentro de uno de los bucles internos.
- El tamaño del triángulo lo elige el usuario al principio, pero si es muy grande puede visualizarse mal.
- Observe que la condición de parada de los bucles depende de n (tamaño del triángulo, número de filas) y de la variable i, que indica el número de fila que está siendo procesada en cada momento (variable del bucle principal).

En la Figura 4.9 se puede apreciar una imagen estática de este vídeo en la cual se explica y resuelve cómo pintar un triángulo relleno por caracteres en pantalla.

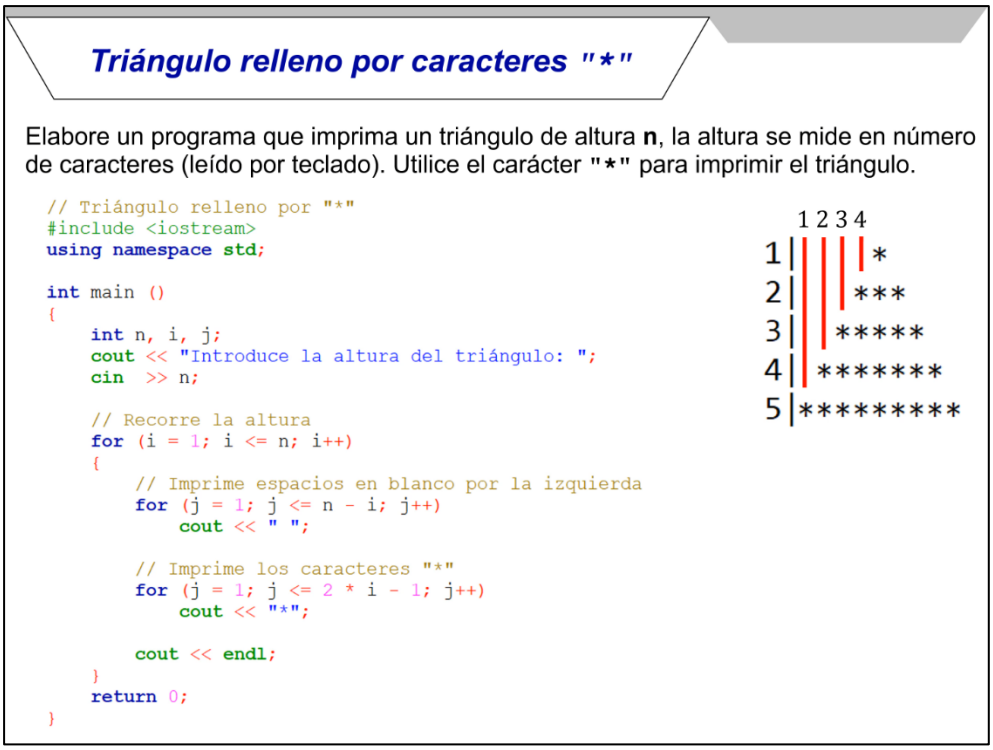

Figura 4.9: Una imagen de ejemplo del vídeo: "Dibujar un triángulo de asteriscos (relleno y hueco). Fuente: Anexo 1.

- **Código:**

```
// Imprime un Triángulo hueco en pantalla

#include <iostream>
using namespace std;

int main ()
{
    int n, i, j;
    cout << "Introduce la altura del triángulo: ";
    cin  >> n;
    // Recorre la altura
    for (i = 1; i <= n - 1; i++)
    {
        // Imprime espacios en blanco por la izquierda
        for (j = 1; j <= n - i; j++)
            cout << " ";

        // Imprime los caracteres "*" y " "
        for (j = 1; j <= 2 * i - 1; j++)
            if (j == 1 || j == 2 * i - 1)
                cout << "*";
            else
                cout << " ";

        cout << endl;
    }
```

```
    // Imprime "*" última fila
    for (i = 1; i <= 2 * n - 1; i++)
        cout << "*";
    return 0;
}
```

4.3.4. Vídeo: Programa C/C+ para analizar un Circuito Eléctrico simple con 3 resistencias en serie o en paralelo

En este vídeo se explica un programa sencillo que calcula la intensidad de corriente de un circuito eléctrico con 3 resistencias conectadas en serie o en paralelo:

- Se hace un menú para que el usuario pueda meter y cambiar los datos con facilidad.
- Se comprueban algunos posibles errores.
- Se explican los conceptos básicos del bucle do-while y de la sentencia switch. Ambas sentencias se explican con detalle en otros vídeos propios para cada una de ellas.

En la Figura 4.10 se puede apreciar una imagen estática de este vídeo en la cual se explica cómo crear el menú requerido por el problema, así como una breve explicación del bucle do-while.

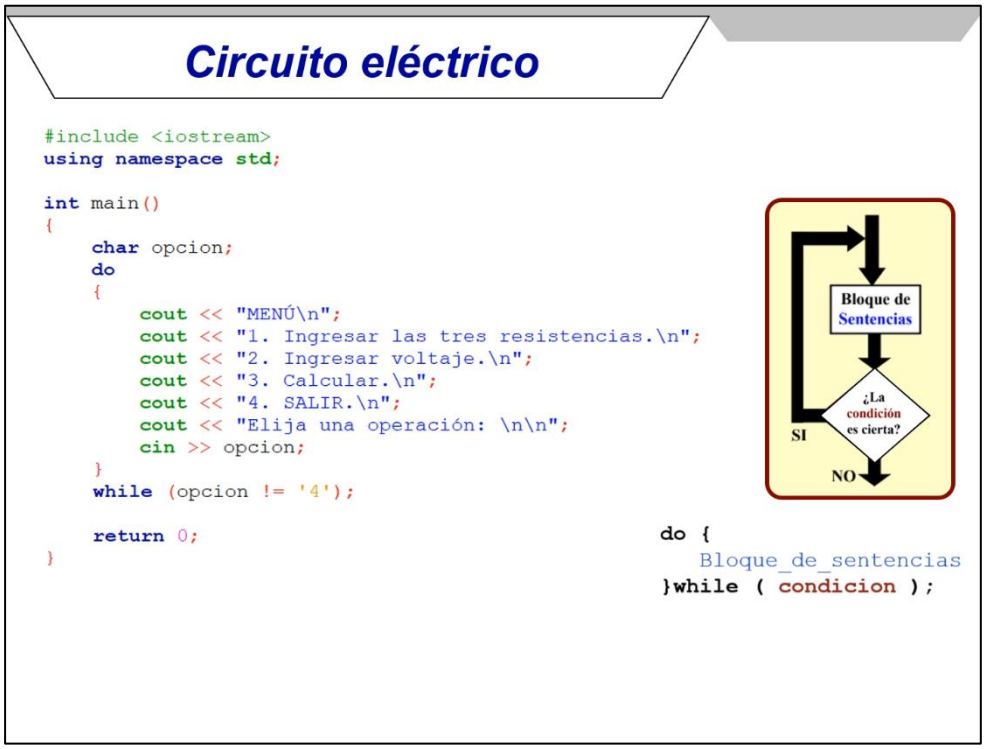

Figura 4.10: Una imagen de ejemplo del vídeo: "Programa C/C++ para analizar un Circuito Eléctrico simple con 3 resistencias en serie o en paralelo. Fuente: Anexo 1.

- **Código:**

```
//Calculadora de circuito eléctrico en serie y paralelo
#include <iostream>
using namespace std;
```

```
int main()
{
    char opcion, tipo;
    float R1, R2, R3, R123, V, I;
    bool condicion1 = false, condicion2 = false;

    do {
        cout << "\nMENÚ\n----\n";
        cout << "1. Ingresar las tres resistencias.\n";
        cout << "2. Ingresar voltaje.\n";
        cout << "3. Calcular.\n";
        cout << "4. SALIR.\n";
        cout << "\nElija una operación: ";
        cin >> opcion;

        switch(opcion) {
            case '1':
                cout <<"\n* Escoja el tipo de conexión:\n";
                cout << "1) Resistencias en SERIE. \n";
                cout << "2) Resistencias en PARALELO.\n";
                cin >> tipo;
                if (tipo == '1' || tipo == '2')
                {
                    cout << "\nIngrese el valor de R1
                    (Ohmios): ";
                    cin >> R1;
                    cout << "Ingrese el valor de R2: ";
                    cin >> R2;
                    cout <<"Ingrese el valor de R3: ";
                    cin >> R3;
                    condicion1 = true;
                }
                else
                {
                    cout <<"\nIncorrecto."<<endl;
                    condicion1 = false;
                }
                break;
            case '2':
                    cout << "\nIngrese el voltaje de la fuente
                    (Voltios): ";
                    cin >> v;
                    condicion2 = true;
                break;
            case '3':
                    if(condicion1 && condicion2)
                    {
                        if (tipo == '1')
                        {
                            // Resistencias en serie
                            R123 = R1 + R2 + R3;
                            I = V/R123; // Si R123 = 0 -> ERROR
                            cout << "\nRESISTENCIA total en
                            serie : " << RTSerie << endl;
                            cout << "INTENSIDAD del circuito
```

```
                                con resistencias en serie es: " <<
                                I << endl;
                            }
                            else
                            {
                                // Resistencias en paralelo
                                R123 = 1/((1/R1)+(1/R2)+(1/R3));
                                I = V/R123; // Si R123 = 0 -> ERROR
                                cout << "\nRESISTENCIA total en
                                paralelo: " << RTParalelo << endl;
                                cout << "INTENSIDAD del circuito
                                con resistencias en paralelo es: "
                                << I << endl;
                            }
                        }
                        else
                        {
                            if(!condicion1)
                                cout << "\nERROR: Tiene que
                                ingresar   el   valor   de   las
                                RESISTENCIAS.\n";
                            if(!condicion2)
                                cout << "\nERROR: Tiene que
                                ingresar el valor del voltaje.\n";
                        }
                    break;
                case '4':
                    cout << "Fin del programa." << endl;
                    break;
                default:
                    cout << "Error. Has introducido una opción
                    incorrecta. " << endl;
            }
        }
        while(opcion!='4');
        return 0;
}
```

4.4. Tema 5. Funciones

En este tema se desarrollan los subprogramas y funciones. En lenguaje C/C++, una función es un bloque de código que se puede llamar tantas veces como se desee desde otras partes del código en un programa. Las funciones son útiles porque permiten dividir un programa en subprogramas más pequeños y manejables, y también porque permiten reutilizar código de forma cómoda.

Para definir una función en C/C++, debes especificar el tipo de datos de retorno de la función, el nombre de la función y los parámetros de entrada/salida (si los hay). Aquí hay un ejemplo de cómo se define una función que suma dos números enteros y devuelve el resultado:

```
int suma(int x, int y) {
  return x + y;
}
```

Para llamar a una función, simplemente escribes su nombre seguido de los argumentos entre paréntesis. Por ejemplo:

```
int a = 3, b = 4;
int c = suma(a, b);  // c contiene el valor 7
```

En el ejemplo anterior se usa el paso de argumentos por valor, que se usa exclusivamente para argumentos de entrada. Por otra parte, el paso de argumentos por referencia se usa para argumentos de entrada/salida, o bien, solo de salida. Por último, el paso de argumentos por referencia constante se usa también solo para argumentos de entrada. En otros vídeos ya existentes en el canal de YouTube se explica tanto la creación de funciones como los diferentes tipos de pasos de argumentos que existen.

4.4.1. Vídeo: Programa C/C++ para calcular la distancia euclídea entre 2 puntos 2D, con funciones

En este vídeo se explica un programa sencillo que lee dos puntos en el plano cartesiano, con dos coordenadas cada punto (x,y) y calcula la distancia euclídea entre ellos:

- La lectura, el cálculo y la escritura del resultado se hacen en funciones independientes.
- Se explica brevemente el paso de argumentos por valor y por referencia.
- Se explica brevemente cómo se deben situar las funciones en el programa final, con respecto a la función main().
- Se explica brevemente distintas formas de organizar la función principal con las 3 funciones previamente definidas.

En la Figura 4.11 se puede apreciar una imagen estática de este vídeo en la cual se resuelven y explican las funciones necesarias para resolver el problema.

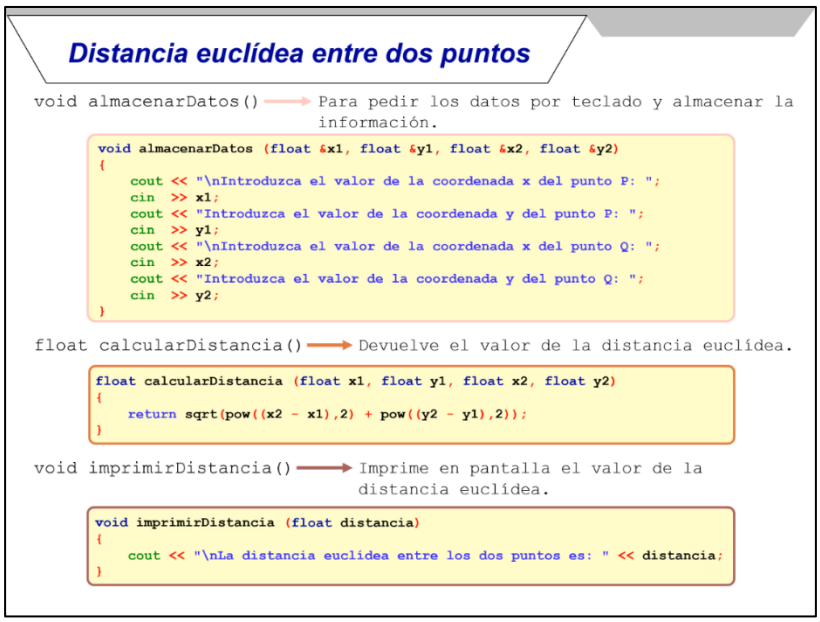

Figura 4.11: Una imagen de ejemplo del vídeo: "Programa C/C++ para calcular la distancia euclídea entre 2 puntos 2D con funciones". Fuente: Anexo 1.

- **Código:**

```
// Hallar la distancia euclídea entre dos puntos

#include <iostream>
#include <math.h>
using namespace std;

void almacenarDatos (float &x1, float &y1, float &x2, float &y2);
float calcularDistancia (float x1, float y1, float x2, float y2);
void imprimirDistancia(float distancia);

int main ()
{
    float x1, x2, y1, y2;

    almacenarDatos (x1,y1,x2,y2);
    imprimirDistancia(calcularDistancia(x1,y1,x2,y2));

    return 0;
}

// Almacena los datos por referencia
void almacenarDatos (float &x1, float &y1, float &x2, float &y2)
{
    cout << "\nIntroduzca el valor de la coordenada x del punto
    P: ";
    cin  >> x1;
    cout << "Introduzca el valor de la coordenada y del punto
    P: ";
    cin  >> y1;
    cout << "\nIntroduzca el valor de la coordenada x del punto
    Q: ";
    cin  >> x2;
    cout << "Introduzca el valor de la coordenada y del punto
    Q: ";
    cin  >> y2;
}

// Calcula la distancia euclídea entre dos puntos
float calcularDistancia (float x1, float y1, float x2, float y2)
{
    return sqrt(pow((x2 - x1),2) + pow((y2 - y1),2));
}

// Imprime en pantalla el valor de la distancia euclídea
void imprimirDistancia (float distancia)
{
    cout << "\nLa distancia euclídea entre los dos puntos es:
    " << distancia;
}
```

4.4.2. Vídeo: Programa C/C++ para comprobar la Conjetura de Goldbach

En este vídeo se explica un programa sencillo que ayuda a comprobar la Conjetura de Goldbach para todos los números en un determinado rango [a,b]. Es decir, si todo número par mayor que 2 dentro de ese rango puede escribirse como suma de dos números primos:

- Se explica una función para comprobar si un número es o no primo.
- Se expone un bucle que recorre todos los números pares dentro del intervalo de entrada [a,b].
- Se buscan todas las posibles parejas de números que suman cada valor par del rango, y se comprueba si ambos números son primos.
- Recuerde que el número 1 no se considera primo porque solo tiene 1 divisor (el propio 1), y los números primos deben tener 2 divisores (el 1 y el propio número).
- En realidad, el programa mostrado NO nos dice si se cumple o no la conjetura de Goldbach, sino que solo nos muestra los números pares que se pueden formar como suma de dos primos. El usuario debe comprobar si en la lista de números mostrada en pantalla están, o no, todos los números pares del rango. Se deja como ejercicio que esa comprobación la haga también el programa, mostrando al final un mensaje que indique si la conjetura se cumple o no para el intervalo [a,b].

En la Figura 4.12 se puede apreciar una imagen estática de este vídeo en la cual se muestra y explica el enunciado del problema que se va a resolver en el propio vídeo.

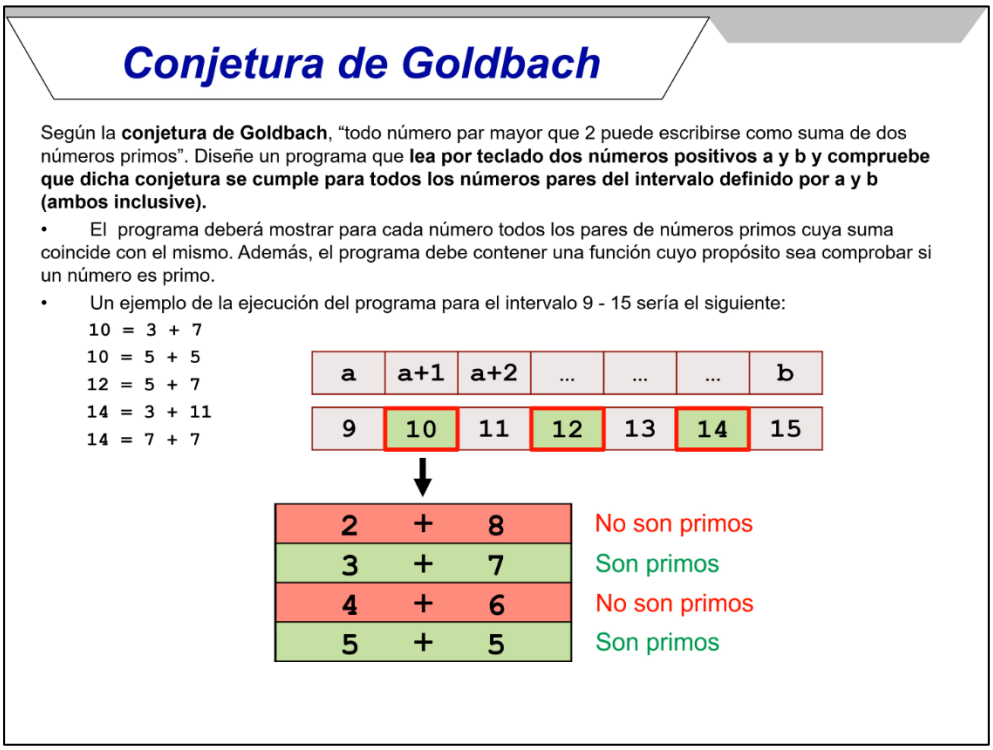

Figura 4.12: Una imagen de ejemplo del vídeo: "Programa C/C++ para comprobar la Conjetura de Goldbach". Fuente: Anexo 1.

- **Código:**

```
// Calcular la suma de dos números primos según la conjetura
de Goldbach

#include <iostream>
using namespace std;

bool EsPrimo (int numero);

int main ()
{
    int a, b, i, n1, n2;
    do
    {
        cout << "Introduzca el valor de a y b: ";
        cin  >> a >> b;
    }
    while (a >= b); // Forzamos que a sea menor que b

    if (a%2 != 0) // Inicio el intervalo en un número par
        a++;

    for (i = a; i <= b; i += 2) // Bucle de números pares
    dentro del intervalo
        for (n1 = 2; n1 <= i/2; n1++) // Descomposición del
         número par
        {
            n2 = i - n1;
            if (EsPrimo(n1) && EsPrimo(n2 )) // Si ambos son
            números primos se imprimen los valores
                cout << i << " = " << n1 << " + " << n2 <<
                endl;
        }
}

bool EsPrimo (int numero) // Devuelve true si el número es
primo
{
    int i, CuentaDivisores  = 2 ;
    for (i = 2 ; i < numero / 2 ; i++)
        if (numero%i == 0)
            CuentaDivisores++;

    if (CuentaDivisores == 2)
        return true;

    return false;
}
```

4.5. Tema 6. Datos Estructurados

Este es el último tema de la asignatura, y engloba todos los conceptos vistos en la misma. Además, amplía la información ya vista con nuevos conceptos sobre datos estructurados. Los datos estructurados son tipos de datos que permiten almacenar varios valores diferentes en una sola variable. Hay varias formas de crear datos estructurados en C++, como las estructuras, las clases y los arrays. El concepto de clase es propio de programación orientada a objetos y no está incluido en la asignatura objeto de este TFG.

Una estructura es un tipo de dato que permite agrupar diferentes variables de cualquier tipo en una sola unidad. Por ejemplo, si quisieras almacenar información sobre una persona, podrías usar una estructura para almacenar su nombre, edad y dirección:

```
struct persona {
  string nombre;
  int edad;
  string direccion;
};
```

Una vez hecha la declaración anterior, dentro de una función podemos declarar variables de ese tipo, asignarle valores y, por supuesto, utilizarlos. Unos ejemplos son los siguientes:

```
persona p1;           // declara una variable de tipo "persona"
p1.nombre = "Juan";   // asigna valores a los campos de la estructura
p1.edad = 30;
p1.direccion = "Calle Falsa 123";
```

Un array es una colección finita de elementos del mismo tipo de datos. Para declarar un tipo array necesitamos: incluir la biblioteca estándar de C++ (`<array>`), declarar tanto el tipo de datos como el número de elementos que conforma nuestro array (este debe ser una constante conocida en tiempo de compilación) y por último nombrarlo (el nombre del tipo de datos suele empezar por T mayúscula). Un ejemplo de un array para números enteros de tamaño 5 sería:

```
#include <array>
const unsigned TAMA = 5;
typedef array <int, TAMA> Tvector;
```

4.5.1. Vídeo: Programa C/C++ sencillo para hallar el Centro de masas de un conjunto de partículas en 3D

En este vídeo se explica cómo almacenar los datos de un conjunto de partículas, en la que cada partícula tiene una posición en 3 dimensiones (x,y,z) y una masa. Además, el programa muestra el centro de masas (baricentro) del conjunto:

- Se define un tipo de datos struct y un array de ese tipo.
- Se introducen datos en el array y, en el mismo bucle, se van calculando los valores que necesitamos para calcular el centro de masas.
- Es un ejemplo tan sencillo que no requiere usar arrays, ya que los datos de cada partícula solo los usamos una vez, justo en cuanto son leídas.

- Recomendamos modificar este programa para hacer funciones independientes para leer y escribir datos del tipo "lista de partículas" y, también una función independiente para calcular su centro de masas, sin escribirla. La función principal debería servir para probar esas 3 funciones y otras que pueda imaginar.

En la Figura 4.13 se puede apreciar una imagen estática de este vídeo en la cual se muestra y explica el enunciado del problema que se va a resolver en el propio vídeo y un gráfico 2D con unos ejemplos que ayudan a entender el significado.

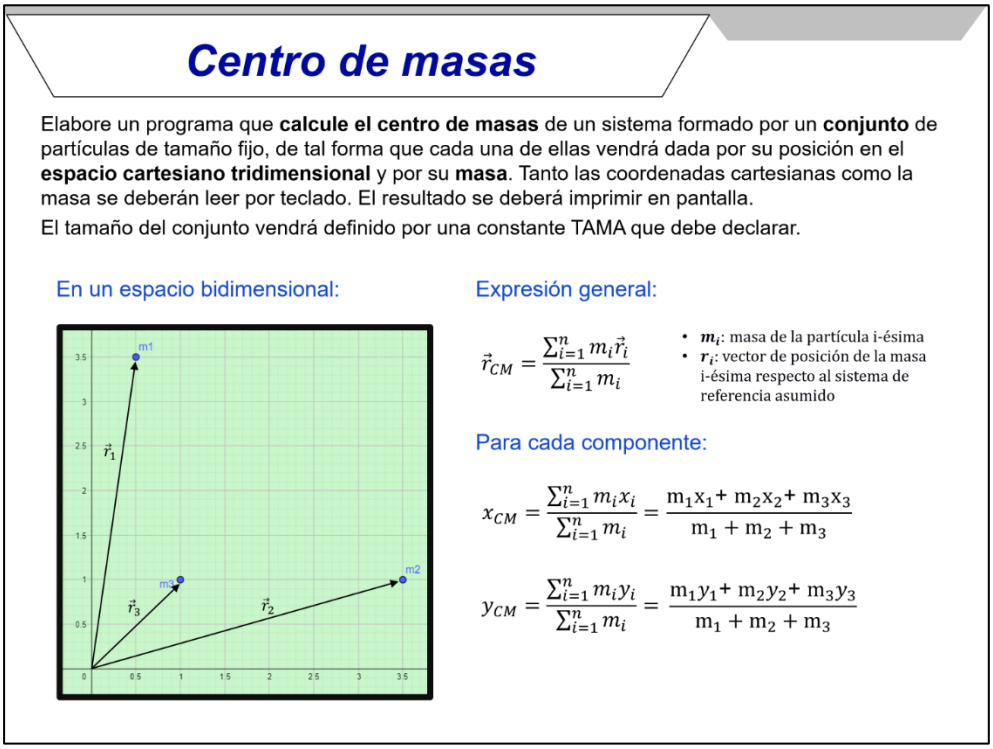

Figura 4.13: Una imagen de ejemplo del vídeo: "Programa C/C++ para hallar el Centro de masas de un conjunto de partículas en 3D". Fuente: Anexo 1.

- **Código:**

```
/*
Programa que calcula el centro de masas de un sistema formado
por un conjunto de partículas de tamaño fijo.
Cada partícula viene dada en sus coordenadas cartesianas y por
su masa
*/

#include <iostream>
#include <array>

using namespace std;

const unsigned TAMA = 3;
```

```
// Definición de una partícula
typedef struct
{
    float x, y, z, masa;
} Tparticula;

// Lista de todas las partículas
typedef array <Tparticula, TAMA> TListaParticulas;

int main ()
{
    int i;
    float X = 0, Y = 0, Z = 0, MasaTotal = 0;
    TListaParticulas T;

    for (i = 0; i < TAMA; i++) // Pedir y almacenar datos
    {

        cout << "\nIntroduce coordenada X de la partícula " <<
        (i+1) << ": ";
        cin  >> T[i].x;
        cout << "Introduce coordenada Y de la partícula: " <<
        (i+1) << ": ";
        cin  >> T[i].y;
        cout << "Introduce coordenada Z de la partícula: " <<
        (i+1) << ": ";
        cin  >> T[i].z;
        cout << "Introduce la masa de la partícula " << (i+1)
        << ": ";
        cin  >> T[i].masa;

        // Conforme se leen los datos se va sumando la masa y
        los productos de masa por las 3 coordenadas
        X += (T[i].x * T[i].masa);
        Y += (T[i].y * T[i].masa);
        Z += (T[i].z * T[i].masa);
        MasaTotal += T[i].masa;
    }

    // Imprimir datos
    cout << "\nLas coordenadas del centro de masa son: ";
    cout << "\n     X: " << (X / MasaTotal);
    cout << "\n     Y: " << (Y / MasaTotal);
    cout << "\n     Z: " << (Z / MasaTotal);
}
```

4.5.2. Vídeo: Criba de Eratóstenes en lenguaje C/C++ para hallar números primos

En este vídeo se explica cómo implementar el método de la criba de Eratóstenes para descubrir los números primos menores a un valor K inicial:

- Primero se explica este legendario algoritmo.

- Se define un array de elementos de tipo bool para indicar si un número está "tachado" o no. Los números que al final no estén "tachados" serán números primos. Nota: El array no emplea las dos primeras posiciones. Desperdiciamos un poco de memoria a cambio de ganar claridad y sencillez en el algoritmo. Tampoco se emplean las últimas posiciones cuando K es menor que el valor máximo.
- Se utilizan bucles anidados, concepto que se explica en otro vídeo.
- Recomendamos modificar este programa para hacer funciones independientes para las distintas operaciones, básicamente calcular los números primos, y escribirlos.

En la Figura 4.14 se puede apreciar una imagen estática de este vídeo en la cual se explica el algoritmo para quitar los números “no primos” de la lista de números que sí cumple el algoritmo de la criba de Eratóstenes.

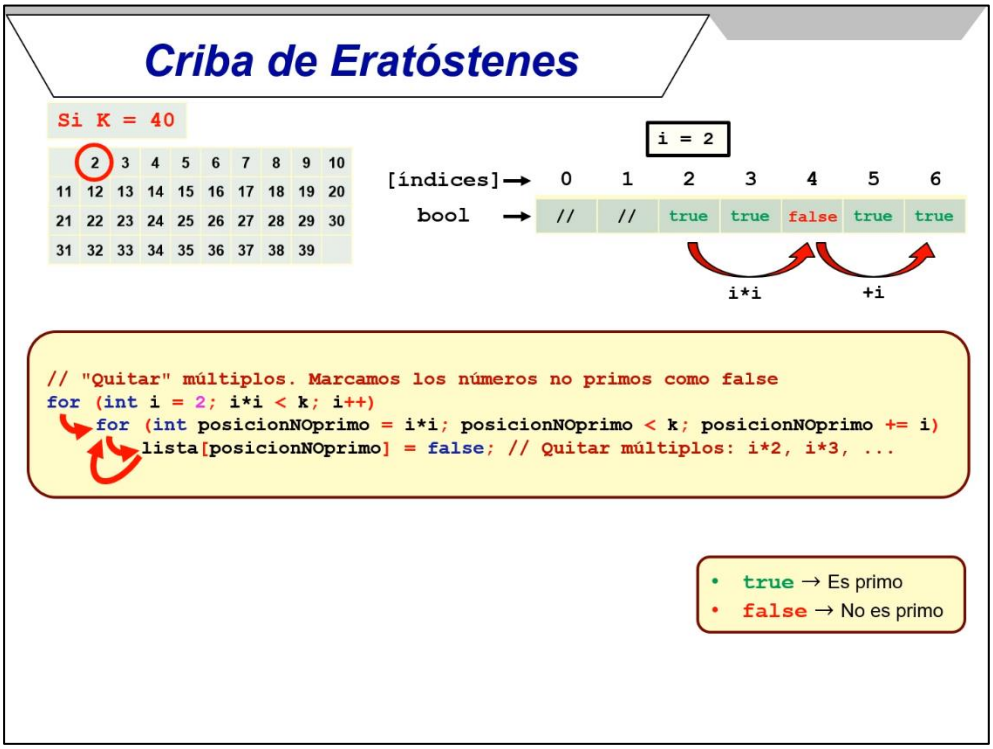

Figura 4.14: Una imagen de ejemplo del vídeo: “Criba de Eratóstenes en lenguaje C/C++ para hallar números primos”. Fuente: Anexo 1.

- **Código:**

```
// Calcular los primos menores a K usando la criba de Eratóstenes
#include <iostream>
#include <array>
using namespace std;

const int TAMA = 1000;
typedef array <bool, TAMA> TPrimos;
```

```
int main ()
{
    TPrimos lista; // Array para los números primos
    int i, j, K, posicionNOprimo;
    do
    {
        cout << "Introduzca el límite superior (menor que
         1000): ";
        cin  >> K;
    }
    while(K >= TAMA);

    // Inicializar intervalo de la criba
    for (i = 2; i < K; i++)
        lista[i] = true;

    // "Quitar" múltiplos. Marcamos los números no primos como false
    for (i = 2; i*i < K; i++)
        if (lista[i]) // Si i es primo, eliminamos sus
         múltiplos
             for (posicionNOprimo = i*i; posicionNOprimo < K;
             posicionNOprimo+=i)
                lista[posicionNOprimo] = false;

    // Mostrar criba. Mostramos los números primos (true)
    cout << "Los números primos menores a " << K << " son: ";
    for (i = 2; i < K; i++)
        if (lista[i])
            cout << " " << i << " ";

    cout << endl << i;
    return 0;
}
```

CAPÍTULO 5.

CONCLUSIONES Y TRABAJOS FUTUROS

En este último capítulo se presentan las conclusiones recabadas a lo largo de la duración del presente TFG. Se dará una visión del esfuerzo que ha supuesto, así como de las experiencias vividas. Se resumirá lo que ha aportado tanto personalmente como al colectivo al que va dirigido, ventajas y desventajas encontradas y posibles ampliaciones y mejoras para seguir expandiendo el contenido del canal de YouTube "*Aprende conmigo*" [11].

5.1. Conclusiones

Como conclusión principal, nos gustaría destacar que el haber creado vídeos sobre programación en lenguaje C/C++ ha sido una forma efectiva de afianzar y expandir los conocimientos que ya tenía el autor sobre la materia. Además, sirve como contenido divulgativo para que otros puedan aprender acerca de la programación y conceptos técnicos. Por tanto, este trabajo no solo servirá para finalizar una titulación, sino que sus resultados serán utilizados durante bastantes años.

Algunas ventajas que presenta el utilizar un contenido multimedia para la divulgación de conocimiento pueden ser:

- **Facilidad de acceso**: los vídeos pueden ser vistos en línea en cualquier momento y lugar, lo que los hace convenientes para estudiar al ritmo establecido por uno mismo. Por supuesto, hace falta un dispositivo con acceso a Internet.

- **Mayor comprensión**: los vídeos pueden proporcionar explicaciones visuales y demostraciones de cómo funcionan las cosas, lo que facilita la comprensión respecto a la lectura de documentos técnicos en el que expliquen el mismo contenido.

- **Mayor retención**: al ver y oír las explicaciones, es más probable que el estudiante se acuerde de lo que ha aprendido. Para ello, se ha pretendido hacer los vídeos con una duración de 5 a 7 minutos. De esta forma, el espectador podrá consumir fácilmente el contenido y es más probable que retenga más información.

Sin embargo, también hay desafíos a la hora de crear contenido multimedia sobre programación, como puede ser asegurarse de que el contenido sea accesible y de fácil comprensión para el público objetivo o mantener un contenido de calidad y actualizado. En general, crear vídeos

sobre programación puede ser una herramienta valiosa tanto para aprender como para enseñar programación, siempre y cuando se aborde de manera adecuada.

En la Tabla 5.1 se desglosa el tiempo que ha sido necesario para llevar a cabo todas las actividades necesarias para la creación del presente TFG. El trabajo de tutorización se refiere solo a lo relacionado con los vídeos directamente, principalmente revisar tanto los guiones escritos de cada vídeo como las diferentes versiones hasta alcanzar la versión final, además de publicar cada vídeo en YouTube. A título orientativo, ello supone unos 15 minutos de trabajo de tutorización por cada minuto de vídeo definitivo. En la tutorización no se ha incluido el tiempo de revisión de otras tareas del TFG, tales como todo lo relacionado con la memoria.

Concepto	Tiempo invertido (horas)
Investigación y documentación	28
Redactar los guiones	35
Crear las presentaciones	80
Grabar el audio	14
Editar las animaciones y cuadrar el audio	98
Tutorización	16
Realización de la memoria	40
TOTAL	**313**

Tabla 5.1: Relación tareas-tiempo invertido en la realización del presente TFG. Fuente: Elaboración propia

Si analizamos el tiempo invertido de todas las actividades que se han necesitado para la creación de los vídeos (ignorando el tiempo dedicado a la realización de la memoria), hemos comprobado que tardamos **212,18 minutos en producir 1 minuto de vídeo definitivo**.

$$\frac{Tiempo\ total\ invertido\ en\ la\ creación\ de\ los\ vídeos\ (Tabla\ 1)}{Duración\ total\ de\ los\ vídeos\ creados\ (Tabla\ 2)}$$

$$= \frac{313 \times 60\ minutos\ de\ trabajo}{88{,}51\ minuto\ de\ vídeo\ definitivo} = 212{,}18\ \frac{minutos\ \text{de trabajo}}{minuto\ de\ vídeo\ definitivo}$$

5.2. Aprendizaje

En cuanto a las competencias y aptitudes adquiridas por el autor a lo largo del trabajo hay que destacar principalmente la creación y edición de vídeos. A lo largo del grado, el autor ya había utilizado previamente la herramienta PowerPoint para crear presentaciones para exponer trabajos de diversas asignaturas, pero en este TFG se han utilizado funciones y configuraciones totalmente nuevas e inexploradas previamente. Además, de la necesidad de crear un contenido dinámico y entretenido para el espectador, se ha mejorado la forma de representar la información en pantalla. Sabiendo identificar los puntos clave y destacables y como resaltarlos de manera natural y efectiva.

Otra cualidad adquirida gracias a este trabajo ha sido obtener una visión más profunda en cuanto a conocimientos sobre programación, siendo necesarios un profundo conocimiento y entendimiento para ser capaz de poder reflejarlo y explicárselo al espectador de forma que este sea capaz de entenderlo.

Por último, se han mejorado las competencias narrativas del autor gracias a redactar los guiones de los vídeos. Esto surge de la necesidad de explicar diversos conceptos de programación y resoluciones de ejercicios de una manera clara y concisa. Además, al tener que comunicar conceptos técnicos de programación de manera comprensible para un amplio público, también se han mejorado las habilidades para transmitir ideas complejas de forma efectiva.

5.3. Objetivos cumplidos

El objetivo principal de este trabajo era continuar con la creación de contenido multimedia para el canal de YouTube "*Aprende conmigo"* dirigido y editado por el tutor de este TFG.

Para ello, se han creado un total de 14 vídeos de carácter entretenido y formativo que incluyen conceptos teóricos y, principalmente, la resolución de problemas de programación en C/C++ relacionados con diversos campos de la ingeniería. Cabe recordar que se ha seguido el estilo y formato de los vídeos ya encontrados en el canal previamente. De esta forma, no se apreciarán cambios significativos en el formato audiovisual, lo que facilita que se pueda mantener el mismo formato para futuros vídeos del canal.

En la tabla 5.2 se expone a modo de resumen los vídeos creados con la duración en minutos para cada uno. La duración total es de **1 hora con 22 minutos y 31 segundos** de contenido multimedia. El tiempo medio de los 14 vídeos es de 5 minutos con 53 segundos.

5.4. Ventajas y desventajas encontradas

Durante el proceso de creación de este TFG el autor ha observado ciertas ventajas y desventajas que podemos encontrar en la realización y uso del material creado.

Una de las principales ventajas a la hora de crear los vídeos, han sido las herramientas utilizadas. Todas ellas han sido recomendadas por el propio tutor en base a su experiencia previa creando y editando los vídeos presentes en el canal "Aprende conmigo". Estas herramientas son fáciles e intuitivas de utilizar. Además, casi todas se encuentran disponibles en Internet para poder descargarlas e instalarlas. El único problema puede surgir debido a PowerPoint, ya que es el único software de pago de las herramientas utilizadas, pero esto se soluciona fácilmente utilizando las credenciales personales del alumno de la Universidad de Málaga para poder instalarlo. Esto se debe a que la Universidad de Málaga pone a disposición de sus estudiantes una licencia para ellos de las herramientas de Microsoft 365. Para poder descargar estas herramientas y otras más solo hay que seguir las instrucciones que se encuentran dentro de la propia página web [20].

Tema	Título vídeo	Carácter	Tiempo (min)
3	Operaciones básicas del tipo string en C/C++	Teórico	5:19
3	Operadores ++ y — en C/C++ (preincremento, postincremento, predecremento y postdecremento)	Teórico	5:27
3	Operaciones simples entre enteros y reales (tipos de datos int, float, double y long double)	Práctico	7:09
3	Programa C/C++ simple sobre el tipo string	Práctico	4:32
3	Programa C/C++ para ver el código de un carácter (molde int) y su tamaño en bytes (operador sizeof)	Práctico	4:15
3	Programa C/C++ para resolver una ecuación de segundo grado	Práctico	6:20
4	Programa C/C++ para contar cuántas veces aparece un carácter en una frase	Práctico	6:06
4	Programa C/C++ para calcular las raíces de una función matemática por el método de Newton-Raphson	Práctico	6:28
4	Dibujar un triángulo de asteriscos (relleno y hueco)	Práctico	5:34
4	Programa C/C+ para analizar un Circuito eléctrico simple con 3 resistencias en serie o en paralelo	Práctico	6:58
5	Programa C/C++ para calcular la distancia euclídea entre 2 puntos 2D, con funciones	Práctico	4:19
5	Programa C/C++ para comprobar la Conjetura de Goldbach	Práctico	5:52
6	Programa C/C++ sencillo para hallar el Centro de masas de un conjunto de partículas en 3D	Práctico	6:28
6	Criba de Eratóstenes en lenguaje C/C++ para hallar números primos	Práctico	7:32
TOTAL			**88:51**

Tabla 5.2: Resumen de los vídeos creados para cada tema con su duración en minutos. Fuente: Elaboración propia

Precisamente, uno de los principales inconvenientes encontrados durante el proceso de creación de este trabajo tiene que ver con PowerPoint. Esta herramienta no está diseñada para ser un editor de vídeos completo, lo que significa que el tiempo requerido para crear un vídeo está directamente relacionado con el número de elementos y animaciones utilizados. Cuanto más elementos visuales se incluyan en la presentación, más animaciones se requieren (que suelen ser al menos dos por elemento, una para el efecto de entrada y otra para el de salida), y por tanto más tiempo y esfuerzo se necesitará para trabajar con ellos y ajustar los tiempos de las animaciones. Esto puede resultar extremadamente tedioso e ineficiente. En la Figura 5.1 se muestra una diapositiva en PowerPoint para un vídeo, donde se aprecia la cantidad de elementos que hay en la misma (columna de la derecha: panel de selección) y las animaciones asociadas a cada elemento (rectángulos pequeños y enumerados distribuidos a lo largo de la imagen).

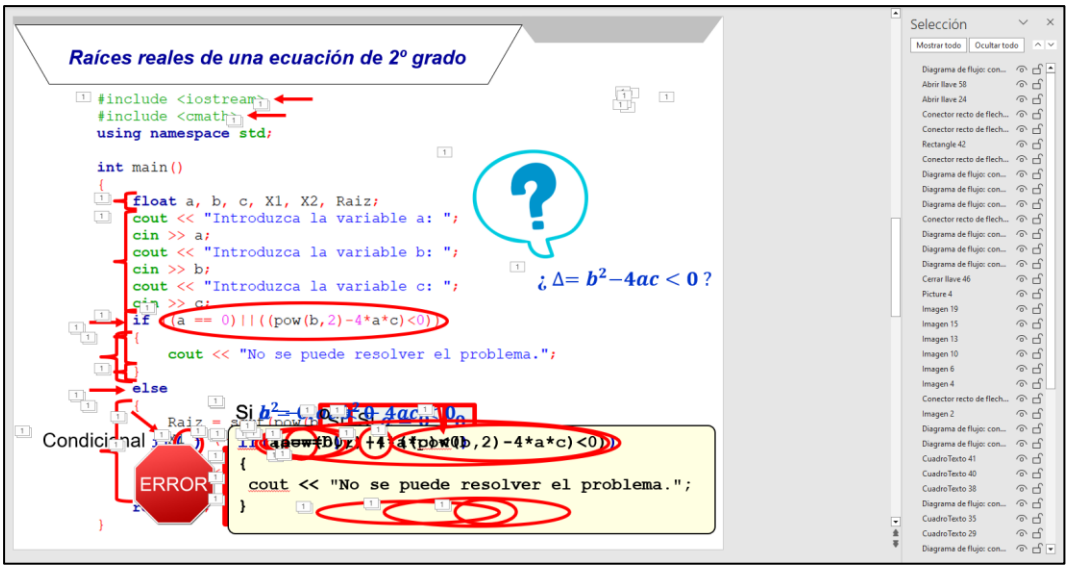

Figura 5.1: Ejemplo de diapositiva de un vídeo en PowerPoint. Fuente: Elaboración propia

Como se muestra en la Tabla 5.1, la mayor cantidad de tiempo invertido en este trabajo fue en cuadrar las animaciones con el audio, que requirió de un total aproximado de 98 horas. Esto se debe en parte a que la única forma de comprobar si el resultado es el deseado es: reproducir la presentación desde la diapositiva en cuestión. Por ejemplo, si se quiere comprobar una animación que comienza en el minuto 3, se tendrá que reproducir los primeros 3 minutos. Este proceso resulta extremadamente ineficiente, sin embargo, en el Capítulo 3. Metodología - Herramientas de composición de vídeos - PowerPoint se propone una solución que puede reducir significativamente el tiempo invertido. Resumidamente, se ha optado por dividir la presentación en varias diapositivas, de esta forma el número de elementos y animaciones que se gestionan es mucho menor. Por lo cual, el tiempo que se invierte en comprobar si las animaciones estan bien se reduce considerablemente.

Este problema se ha analizado y comentado en el Capítulo 3. Se ha optado por esta solución precisamente por la facilidad que tiene el propio software PowerPoint para crear presentaciones y contar una narrativa, que nada tiene que ver con las opciones de edición de vídeos. Otra opción que se ha planteado es utilizar otro software de edición de vídeos, pero este tipo de software requiere una gran cantidad de tiempo para poder manejarlo con soltura y comodidad, por lo que al final el tiempo que sería necesario invertir en dominar el software podría ser mayor al problema en cuestión. También se ha tenido en cuenta seguir utilizando PowerPoint a pesar de sus limitaciones, pues es un software que el autor ya había trabajado previamente y conocía, además que profundizar en este software puede ayudar al autor de cara a realizar futuras presentaciones de trabajo que requieran de este software.

5.5. Posibles mejoras y trabajos futuros

La principal línea de actuación a la que se enfrentaran los futuros alumnos que dediquen su TFG a este proyecto será la creación de nuevos vídeos para el canal "*Aprende conmigo*" en YouTube. Para estos nuevos vídeos, hay que tener en consideración que a fecha de la entrega de este documento se encuentran disponibles 45 vídeos en el canal de YouTube "*Aprende conmigo*", de los cuales solamente 12 son de ejercicios prácticos, mientras que los 33 vídeos restantes son principalmente sobre teoría del temario de la asignatura. Sería interesante seguir profundizando los vídeos de orientación práctica, ya que de esta forma el alumnado podrá tener material que complemente sus conocimientos adquiridos en clase o en los propios vídeos de teoría con ejercicios resueltos y explicados. Aun así, esto no significa que no se deba seguir realizando más material de contenido teórico, ya que siempre se pueden ampliar los conocimientos o conceptos sobre todos los temas e incluso equilibrar la cantidad de vídeos sobre los diferentes temas. E incluso, se podría realizar una encuesta a los propios alumnos sobre qué temas necesitan un mayor aporte de información o cuales están suficientemente desarrollados.

Por ejemplo, una de las lagunas que tiene el canal está en la carencia de explicaciones sobre el compilador, tanto las opciones básicas como otras más avanzadas, como es el depurador.

Otra línea de investigación que podría ser interesante sería analizar la utilidad y éxito de estos vídeos, ya que pueden ser una forma atractiva y efectiva de transmitir conocimientos a los estudiantes. Además, estos vídeos también pueden tener el potencial de promover el interés por la programación entre aquellas personas que todavía no se han adentrado en este campo. Para ello, se podrían emplear los datos que genera YouTube en cuanto a estadísticas, como puede ser el tiempo medio de visualizado real que tiene un vídeo, cuáles son los vídeos que reciben más visitas y por qué, en qué fechas recibe más cantidad de visitas el canal, la localización geográfica de los espectadores, etc.

Este análisis no se ha podido llevar a cabo en este trabajo debido a que los vídeos presentados se han subido hace relativamente poco tiempo, por lo que no se han podido recabar suficientes datos como para poder realizar un análisis efectivo.

Por otro lado, comentar que la creación y edición de vídeos puede suponer una enorme inversión de tiempo. Es por ello, que sería enriquecedor y más eficiente si esta clase de trabajo fuese elaborado en grupos multidisciplinares, estando formado por una persona técnica que tenga suficientes conocimientos para explicar un concepto o resolver un problema de programación, y otro individuo que esté especializado en la creación de contenido multimedia, pudiendo utilizar software de edición y animación más avanzados o con más pericia. De esta forma, la calidad de los vídeos sería superior. E incluso se podría abarcar la creación de más vídeos al ser una forma de trabajo más eficiente.

En resumen, hay muchas líneas de investigación futura interesantes que podrían ser exploradas en un TFG sobre la creación de contenido multimedia de programación en lenguaje C o C++. Por ejemplo, otra línea de investigación interesante es la capacidad de estos vídeos para fomentar la creatividad. Esta es una capacidad humana esencial que nos permite generar nuevas ideas y soluciones a problemas de manera original y efectiva. En el contexto de la programación, puede ser especialmente útil para resolver problemas complejos y encontrar soluciones innovadoras. Además, también afecta en la creación de contenido multimedia, ya que puede ayudar a hacer que el contenido sea más atractivo para los espectadores.

REFERENCIAS

[1] *Almenara, J. C., Batanero, J. M. F., & Graván, P. R. (2010). Edición de vídeo digital para profesores: diseño y producción de materiales educativos. Alianza Editorial.*

[2] *Aprendizaje de PowerPoint para Windows - Soporte de Office*. https://support.microsoft.com/es-es/office/aprendizaje-de-powerpoint-para-windows-40e8c930-cb0b-4

[3] Audacity ® | Free, open source, cross-platform audio software for multi-track recording and editing. (web oficial). https://www.audacityteam.org/

[4] *Biggest social media platforms 2022 | Statista*. (web oficial). https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[5] *Blackboard*. (web oficial). https://www.blackboard.com/es-es

[6] Cifuentes-Faura, J. (2020). Docencia online y Covid-19: la necesidad de reinventarse. Revista De Estilos De Aprendizaje, 13(Especial), 115–127. https://doi.org/10.55777/rea.v13iEspecial.2149

[7] *Code::Blocks*. (web oficial). https://www.codeblocks.org/

[8] *Curso de Audacity - YouTube*. https://www.youtube.com/watch?v=scDv5lIk9_w

[9] Dale, N. B., & Weems, C. (2005). *Programming and Problem Solving with C++. Macmillan Publishers.*

[10] Galindo, J. (2022). *Apuntes y transparencias de la asignatura Fundamentos de Informática para los grados de ingeniería industrial de la Universidad de Málaga*.

[11] Galindo, J. (Ed.). *Canal de YouTube: "Aprende conmigo" – YouTube* https://www.youtube.com/@aprendeconmigoinformatica

[12] Galindo, J., Galindo, P., & Corral, J. M. R. (2019). Multimedia System for Self-learning C/C++ Programming Language. En proceedings of the 23rd International Conference on Emerging Technologies and Applications in Telecommunications, Engineering and Technology (ICETA 2019) (pp. 55-64). https://doi.org/10.1007/978-3-030-36778-7_7

[13] *GitHub*. (web oficial). https://github.com/

[14] *Microsoft PowerPoint Software | Microsoft 365*. (web oficial). https://www.microsoft.com/en-ww/microsoft-365/powerpoint

[15] *Moodle - Open-source learning platform.* (web oficial). https://moodle.org/

[16] Sánchez, P. J. S. (1997). *Ejercicios Resueltos de Programación C. Universidad de Cádiz Servicio de Publicaciones.*

[17] *Stack Overflow*. (web oficial). https://stackoverflow.com/

[18] *Udemy*. (web oficial). https://www.udemy.com/es/

[19] *UNED | Universidad Nacional de Educación a Distancia*. (web oficial). https://www.uned.es/universidad/inicio.html

[20] *Web con software de la Universidad de Málaga*. https://software.uma.es/

[21] *YouTube*. (web oficial). https://www.youtube.com/

ANEXO 1:

VÍDEOS CREADOS Y SUS ENLACES

A.1. Tema 3

Operadores ++ y — en C/C++ (preincremento, postincremento, predecremento y postdecremento) - YouTube. https://www.youtube.com/watch?v=tYB5Fr5FBJ8&list=PLK--SOfNfV0MxRO7v6cynDvVLvLbO3tDi&index=3

Operaciones básicas del tipo string en C/C++ - YouTube. https://www.youtube.com/watch?v=v9dFv2eCObo&list=PLK--SOfNfV0MxRO7v6cynDvVLvLbO3tDi

Programa C/C++ para resolver una ecuación de segundo grado - YouTube. https://www.youtube.com/watch?v=kwP9FSdbn4s&list=PLK--SOfNfV0NaI4470h8dTgitOf4U3lga

Programa C/C++ para ver el código de un carácter (molde int) y su tamaño en bytes (operador sizeof) - YouTube. https://www.youtube.com/watch?v=FwNvARV-93E&list=PLK--SOfNfV0NaI4470h8dTgitOf4U3lga&index=4

Operaciones simples entre enteros y reales (tipos de datos int, float, double y long double) - YouTube. https://www.youtube.com/watch?v=8SLwgBpBaJ0&list=PLK--SOfNfV0NaI4470h8dTgitOf4U3lga&index=5

Programa C/C++ simple sobre el tipo string - YouTube. https://www.youtube.com/watch?v=RDMeDriDx7w&list=PLK--SOfNfV0NaI4470h8dTgitOf4U3lga&index=3

A.2. Tema 4

Dibujar un triángulo de asteriscos (relleno y hueco) - YouTube. https://www.youtube.com/watch?v=09PCJYtyTYM&list=PLK--SOfNfV0NMjmQJ5BbNgof7-VWO5LM0&index=5

Programa C/C+ para analizar un Circuito Electrico simple con 3 resistencias en serie o en paralelo - YouTube. https://www.youtube.com/watch?v=8xHUHymhegg&list=PLK-SOfNfV0NMjmQJ5BbNgof7-VWO5LM0

Programa C/C++ para contar cuántas veces aparece un carácter en una frase - YouTube. https://www.youtube.com/watch?v=AXlpOwYrsMY&list=PLK--SOfNfV0NMjmQJ5BbNgof7-VWO5LM0&index=4

Programa C/C++ para calcular las raíces de una función matemática por el método de Newton-Raphson - YouTube. https://www.youtube.com/watch?v=IRXE7KcLkkA&list=PLK--SOfNfV0NMjmQJ5BbNgof7-VWO5LM0&index=3

A.3. Tema 5

Programa C/C++ para calcular la distancia euclídea entre 2 puntos 2D, con funciones - YouTube. https://www.youtube.com/watch?v=HtSTLq43O0Y&list=PLK--SOfNfV0M-XtQo_sO2G9nbTgj-pLrp

Programa C/C++ para comprobar la Conjetura de Goldbach - YouTube. https://www.youtube.com/watch?v=qKU-YEY_ne8&list=PLK--SOfNfV0M-XtQo_sO2G9nbTgj-pLrp&index=3

A.4. Tema 6

Criba de Eratóstenes en lenguaje C/C++ para hallar números primos - YouTube. https://www.youtube.com/watch?v=NQKVKwtTZ2Q&list=PLK--SOfNfV0NPJ3gQHcRxceCZNmaK8fsl&index=3

Programa C/C++ sencillo para hallar el Centro de masas de un conjunto de partículas en 3D - YouTube. https://www.youtube.com/watch?v=s5A0GVk9wrw&list=PLK--SOfNfV0NPJ3gQHcRxceCZNmaK8fsl