

# El impacto de asistentes basados en IA en la enseñanza-aprendizaje de la programación

Anónimo

Departamento Anónimo  
Universidad Anónima  
XXXXX Ciudad Anónima  
anonimo@xxxxxx.es

## Resumen

En los últimos años, las capacidades de los asistentes de programación basados en Inteligencia Artificial se han incrementado de forma muy notable, habiendo trascendido el ámbito de las Ciencias de la Computación. El impacto que estos sistemas tienen en el campo de la educación y específicamente en la enseñanza de la programación de ordenadores es particularmente significativo porque se trata del dominio en el que posiblemente obtienen sus mejores resultados. De estos avances se han hecho eco medios de comunicación generalistas suscitando un debate muy interesante y pertinente con una pregunta que atraviesa toda la discusión: ¿suponen estos asistentes el fin de la programación de ordenadores de la forma en la que la concebimos actualmente?. En este trabajo se analiza el posible impacto de esta tecnología en el proceso de enseñanza-aprendizaje de la programación en una asignatura de primer curso del grado en Ingeniería Informática. Se relacionan algunos experimentos realizados con *ChatGPT* aplicados a los ejercicios prácticos de programación que se utilizan como prácticas de laboratorio en la asignatura. El trabajo finaliza con la presentación de algunas conclusiones que se derivan de los experimentos realizados así como con el planteamiento de diversas cuestiones que siendo de incuestionable actualidad carecen de respuestas concluyentes en muchos casos.

## Abstract

In recent years, the capabilities of AI-based programming assistants have increased dramatically and have transcended the field of Computer Science. The impact that these systems have on the field of education and in particular on the teaching of computer programming is particularly significant because this is the field in which they possibly achieve their best results. These advances have been covered in the general media, giving rise to a very interesting and pertinent debate

with a question that runs through the whole discussion: do these assistants represent the end of computer programming as we currently conceive it?. This paper analyses the possible impact of this technology on the teaching-learning process of programming in a first year subject of a Computer Engineering degree. Some experiments carried out with *ChatGPT* applied to the practical programming exercises used as laboratory practice in the subject are presented. The work concludes with the presentation of some conclusions derived from the experiments carried out, as well as with the posing of several questions that, although undoubtedly topical, lack conclusive answers in many cases.

## Palabras clave

Programación, Inteligencia Artificial, Asistentes, ChatGPT, Enseñanza, Aprendizaje

## 1. Introducción

La programación de ordenadores es una actividad transversal a cualquier rama de la informática y su importancia es compartida por cualquier especialización en esta titulación. *Informática* (ASIG, de ahora en adelante) es una asignatura de 6 créditos que se imparte en el primer cuatrimestre del primer curso del Grado en Ingeniería Informática en la Escuela Anónima de la Universidad de Anónima. Se trata de la primera asignatura (y única en ese primer cuatrimestre) de perfil eminentemente informático que cursa el alumnado de la titulación y es en ASIG donde los estudiantes toman contacto con la programación y en la que han de aprender los fundamentos de esta materia. Los conceptos objeto de estudio son comunes, con pequeñas variaciones, a cualquier lenguaje de programación orientado a objetos, que es el paradigma estudiado en ASIG. El número de estudiantes de primer curso está en torno a los 250 y los contenidos de la asignatura pueden con-

sultarse en la Guía Docente <sup>1</sup>. de la misma. Al margen de tres temas dedicados a una introducción a los Sistemas Operativos, Redes y Bases de Datos, el grueso de los contenidos (en torno a 12 de las 15 semanas del cuatrimestre) se dedican a introducir al alumnado en la programación, siendo C++ el lenguaje vehicular utilizado para estudiarla. Se trata de una asignatura con una importante proporción de contenidos prácticos en la que cada estudiante recibe 4 horas presenciales de clase a la semana distribuidas del siguiente modo:

- 2 horas dedicadas al estudio de contenidos teóricos
- 1 hora dedicada a la resolución de problemas
- 1 hora de prácticas

Se describe a continuación el tipo de actividades que se desarrolla en cada una de este tipo de sesiones.

Las sesiones destinadas a contenidos teóricos se imparten con el formato de clase expositiva y se dedican, con el uso de transparencias que el alumnado tiene disponibles a través del aula virtual, al estudio de los contenidos de la asignatura. Para cumplir con el requisito del número de créditos que cada estudiante ha de cursar en inglés en la titulación, la mayor parte del material que se utiliza en la asignatura está en ese idioma y se incentiva también el uso del mismo en todas las actividades que se desarrollan, particularmente en todo lo relacionado con la materia de programación. Junto a las transparencias, el alumnado dispone de una serie de pequeños programas, disponibles a través de un repositorio público, que sirven de ilustración a los conceptos que se estudian en las sesiones teóricas. Se recomienda al alumnado estudiar esos programas para afianzar los contenidos expuestos.

En las sesiones de problemas el profesorado utiliza un terminal cuya pantalla se proyecta a toda la clase para solucionar de forma práctica algunos problemas seleccionados, resolviendo al mismo tiempo las dudas que puedan surgir en esa resolución.

La Guía docente establece que por cada hora de trabajo presencial, cada estudiante debería dedicar en promedio 1,5 horas de trabajo autónomo, de modo que se espera unas 6 horas de trabajo autónomo semanal por parte de cada estudiante. A pesar de que el peso de los contenidos prácticos de la asignatura suponen solo un 20 % del total de la calificación de la asignatura, se espera que la mayor parte de ese tiempo se destine al diseño y desarrollo de programas de complejidad creciente conforme avanza el desarrollo del curso, y que se evalúan en las sesiones prácticas.

Cada semana al alumnado se le propone la realización de una práctica, consistente en un cierto número (cinco es un número habitual) de programas en C++

relacionados con algún tema estudiado. Esos programas tienen el propósito de servir de “entrenamiento” para que el alumnado afiance conocimientos a lo largo de la semana de la que disponen para realizar esos ejercicios. La sesión semanal de prácticas se destina a la evaluación de esos conocimientos a través de la realización en el laboratorio de prácticas de ejercicios de programación de complejidad similar a los que han sido propuestos con antelación. En las últimas prácticas de la asignatura los ejercicios de evaluación suelen ser modificaciones o ampliaciones de los que se han propuesto para realizar con antelación.

## 2. Motivación

Más allá del ineludible aprendizaje de los conceptos básicos de la materia, es un hecho bien conocido que la práctica es fundamental para aprender a programar ordenadores. Habitualmente el profesorado asigna problemas de programación al alumnado para ayudarles a adquirir esta destreza, y el incremento de las habilidades de una programadora pasa ineludiblemente por muchas horas de dedicación a la realización de programas de complejidad creciente. Forzando un poco las similitudes podríamos decir que la programación se asemeja a las habilidades para conducir un automóvil: cualquier persona con permiso para conducir puede decir que conduce correctamente, pero acreditar que se es una buena conductora requiere muchas horas de práctica y exposición a situaciones infrecuentes. Del mismo modo cualquier informático dirá que sabe programar pero sus habilidades en esta materia dependerán muy directamente de las horas de práctica que haya dedicado a la misma. Siguiendo esta idea, se recomienda al alumnado de ASIG que realice cuantos ejercicios prácticos sean capaces para, de forma progresiva, ir incrementando sus capacidades como programadoras.

Las prácticas de la asignatura se convierten pues en una oportunidad para que el alumnado mejore destrezas y habilidades que le capaciten para abordar los contenidos de asignaturas de cursos posteriores de la titulación.

La masificación de los grupos de laboratorio de prácticas, con grupos de hasta 20 estudiantes por sesión es la mayor dificultad para la evaluación de esos trabajos prácticos. Esta situación es compartida por muchas otras asignaturas de la titulación que tienen una significativa componente práctica en sus contenidos, de modo que algunas cuestiones que en este trabajo se plantean son comunes a muchas asignaturas e incluso a otras titulaciones.

Una vez realizadas tres prácticas iniciales en las que el alumnado se familiariza con el sistema operativo Linux, con el entorno de máquina virtual en el que desarrollará sus programas y con el editor vim, que es el

<sup>1</sup>Guía Docente de ASIG  
<https://www.anonimo.es/>

que se utiliza inicialmente, todas las prácticas restantes abordan contenidos de programación que cubren los siguientes tópicos:

- Primeros programas y conceptos básicos
- Expresiones y tipos de datos
- Alternativas
- Iteraciones
- Funciones
- Cadenas de texto (`std::string`)
- `std::array` y `std::vector`
- Ficheros
- Introducción a la Programación Orientada a Objetos

Desde hace ya varios cursos en ASIG se viene usando la plataforma *Jutge.org* (*juez*) [9] para la evaluación de las prácticas. *Jutge.org* ha sido desarrollado en la Universidad Politécnica de Catalunya, orientado tanto al profesorado como al alumnado. La plataforma aloja una gran cantidad de problemas (aproximadamente unos 2100) que cubren diversos tópicos incluyendo entre ellos los fundamentos de la programación. Los problemas están perfectamente descritos y contienen un conjunto de tests que el código de usuario ha de verificar. Como una de sus características, *Jutge.org* hace hincapié en el trabajo del alumnado, por lo que resulta especialmente útil para reforzar el enfoque de aprender haciendo, en nuestro caso, programando.

El modo de funcionamiento de *Jutge.org* requiere que cuando un estudiante resuelve un problema suba el código fuente de su solución a la plataforma <sup>2</sup> donde se comprueba que sea correcta pasando los correspondientes tests, de los cuales algunos son públicos y otros privados. A una solución se le asigna el veredicto *AC* (*Accepted*) cuando pasa todos los tests existentes para el problema. En su cuenta de *Jutge.org* cada estudiante dispone de un cuadro de mandos en el que se recopila el número de envíos que ha realizado, el de problemas aceptados y rechazados así como diversos gráficos que muestran la evolución en su trabajo con los problemas de la plataforma.

Este modo de trabajo se aprovecha en las prácticas de ASIG: en cada sesión de evaluación se le pide al alumnado que resuelva un pequeño número de problemas (programas) de *Jutge.org*. La evaluación de la sesión depende no solo del número de problemas resuelto (no suelen ser más de dos o tres) sino de la calidad del código desarrollado. *Jutge.org* comprueba exclusivamente (mediante la comprobación de los correspondientes tests) que el programa evaluado funcione correctamente, mientras que en ASIG se enfatizan otros aspectos del código que nos parecen tanto o más relevantes que el propio funcionamiento del mismo, que obviamente es una condición ineludible. Algunos de

los requisitos que se exigen a los programas de prácticas de ASIG se definen en la Guía de estilo de referencia que se utiliza en la asignatura <sup>3</sup>.

Se relacionan a continuación algunos de los requisitos exigidos:

- Correcto sangrado del código
- Adecuado uso de espacios y signos de puntuación en el código
- Adhesión a las reglas de nombrado de identificadores de variables, funciones, clases, etc.
- Todos los identificadores utilizados (salvo excepciones puntuales) han de ser significativos, evitando el uso de “identificadores de un único carácter”.
- Los programas deben escribirse de forma modular incorporando diversas funciones en la solución.
- Todos los ficheros, funciones y métodos del código han de incluir un breve prólogo con comentarios en formato *Doxygen* exponiendo la información más relevante sobre el elemento (función, clase, fichero, ...) en cuestión.
- La compilación de todos los programas ha de automatizarse mediante el uso de herramientas como `make` o `cmake`.
- Se enfatiza que los parámetros de tipos estructurados que se pasen a una función/método se pasen como referencias, cualificadas como constantes, si procede.
- Se promueve que los métodos definidos en las clases sean *const friendly*.

Estos requisitos se introducen al alumnado de forma progresiva y son parte fundamental de la evaluación de cada programa, una vez satisfecho el requisito de un correcto funcionamiento.

Un problema recurrente en todas las asignaturas que requieren la evaluación de prácticas de programación es la infracción por parte de algunos estudiantes de las reglas de código de conducta que establecen que los trabajos presentados a evaluación han de ser programas originales realizados por sus autores. Ante la dificultad de acreditar fehacientemente esta condición en una sesión de evaluación con un elevado número de estudiantes en el aula de prácticas y con un tiempo tan limitado se ha optado por valorar casi exclusivamente el trabajo que el estudiante realiza *in situ* en la sesión de evaluación, y no los ejercicios que durante la semana ha realizado en calidad de preparación para esa evaluación. Se permite al alumnado en la sesión de evaluación el uso de recursos disponibles en internet (documentación, foros, etc.), puesto que entendemos que el alumnado ha de ser evaluado en condiciones de trabajo similares a las que tendrá en un entorno profe-

<sup>2</sup>*Jutge.org* Home Page <https://jutge.org/>

<sup>3</sup>Google C++ Style Guide  
<https://google.github.io/styleguide/cppguide.html>

sional.

Es en este punto donde queremos pasar a considerar los asistentes de programación basados en Inteligencia Artificial (IA). En junio de 2021, GitHub lanzó *Copilot* [8], un “programador de pares de IA” capaz de generar código en diversos lenguajes a partir de cierto contexto como comentarios, nombres de funciones y código adyacente. *Copilot* se basa en un modelo que se entrena con código abierto [3].

En febrero de 2022 *DeepMind* publicó *Alphacode*, otro sistema basado en IA generativa que puede competir con un humano en la resolución de problemas sencillos de programación. Según resultados publicados en *Science* [7], *Alphacode* gana en un 50 % de ocasiones a humanos en competiciones de resolución de problemas de programación.

El pasado 30 de noviembre, *OpenAI* lanzó *ChatGPT* [11], un *chatbot* interactivo de propósito general basado en GPT-3.5 [6], un modelo lingüístico autorregresivo de tercera generación que utiliza *DeepLearning* para producir textos similares a los escritos por humanos.

Tanto *ChatGPT* como *AlphaCode* son “grandes modelos lingüísticos”, es decir, sistemas basados en redes neuronales que aprenden a realizar una tarea a partir de ingentes cantidades de texto generado por humanos. De hecho, ambos sistemas utilizan prácticamente la misma arquitectura, siendo la principal diferencia entre ellos el conjunto de datos con que son entrenados, lo cual los dirige a diferente tipo de tareas. Posiblemente debido a que se trata de un asistente de propósito general, capaz de discutir sobre materias tan dispares como derecho, filosofía o programación de ordenadores, ha sido el lanzamiento de *ChatGPT* el que ha suscitado un mayor interés, no solo en la comunidad informática sino también en medios de comunicación de carácter generalista [4] atrayendo la atención de más de un millón de usuarios apenas cinco días después de su lanzamiento.

A pesar de que hay un consenso generalizado en la comunidad informática respecto a que, al menos en su estado actual, estos asistentes no van a suponer la desaparición de los programadores de ordenadores [2], también es incuestionable el papel que estos asistentes pueden llegar a tener en la toma de decisiones en diversos campos [5].

En la siguiente sección se presentan algunas experiencias significativas realizadas con *ChatGPT* en el ámbito de los ejercicios de programación de las prácticas de ASIG.

### 3. Experiencias con *ChatGPT*

Aunque el estudio que se ha realizado no es exhaustivo, se ha utilizado *ChatGPT* para tratar de resolver

diversos problemas de *Jutge.org* <sup>4</sup>. Debido a limitaciones de espacio no podemos reseñar todos los experimentos realizados, de modo que se exponen solamente los que consideramos más relevantes o de interés para comprender el modo de funcionamiento del bot y sus capacidades para resolver ejercicios concretos.

Todos los enunciados de problemas de *Jutge.org* están públicamente disponibles en la plataforma <sup>5</sup> y cada uno de ellos cuenta con una especificación clara y precisa así como de un conjunto de tests públicos. Se han descrito los problemas a *ChatGPT* tanto en inglés (idioma en que están descritos la mayoría de los problemas de *Jutge.org*) como en español. Para ello se ha iniciado una nueva sesión con el bot en uno u otro idioma dependiendo del caso. Si bien las soluciones que entrega el sistema no son exactamente las mismas, no se han hallado diferencias significativas en cuanto a la calidad tanto algorítmica como de estilo en las soluciones obtenidas. Para que el bot resuelva cualquiera de los problemas no ha resultado necesario suministrarle información adicional a la que figura en el propio enunciado de *Jutge.org*, habiéndosele entregado al bot la información correspondiente a la salida que el programa debiera entregar para los diferentes tests públicos.

Por otra parte, si no se le especifica explícitamente, las soluciones que produce *ChatGPT* generalmente no siguen reglas de estilo o documentación específicas, pero si en el enunciado del problema se incluyen estas reglas como requisito, la solución que entrega el bot se ajusta a las mismas. Es más, al tratarse de un bot interactivo, si la solución que entrega el sistema no es satisfactoria en algún sentido, se le puede pedir que refactorice el código para cumplir con algún requisito y en ese caso en general el sistema adecúa su respuesta para cumplir con la solicitud.

Consideremos en primer lugar el problema *Fermat's last theorem (1)* para el cual se muestra a continuación la descripción que se entregó inicialmente a *ChatGPT*, copiada casi literalmente de *Jutge.org*:

*A famous theorem of the mathematician Pierre de Fermat, proved after more than 300 years, states that, for any natural number  $n \geq 3$ , there is no natural solution (except for  $x=0$  or  $y=0$ ) to the equation  $x^n + y^n = z^n$*

*For  $n = 2$ , by contrast, there are infinite non-trivial solutions. For instance,  $3^2 + 4^2 = 5^2$ ,  $5^2 + 12^2 = 13^2$ ,  $6^2 + 8^2 = 10^2$ , ...*

*Write a C++ program that, given four natural numbers  $a, b, c, d$  with  $a \leq b$  and  $c \leq d$ , prints*

<sup>4</sup>Todos los programas que se mencionan en este trabajo se han alojado convenientemente documentados en un repositorio privado. Cualquier persona interesada en acceder a esos programas puede solicitarlo por correo electrónico al autor (anonimo@anonimo.es)

<sup>5</sup>Jutge Problems <https://jutge.org/problems/>

Listado 1: Test de primalidad suministrado por *ChatGPT*. Complejidad  $O(\sqrt{n})$

```
/**
 * @brief Function that determines if a given number is prime or not.
 *
 * @param number The number to check for primality.
 * @return true if the number is prime, false otherwise.
 */
bool IsPrime(int number) {
    if (number <= 1) return false; // 1 and 0 are not considered prime.
    if (number <= 3) return true; // 2 and 3 are prime.
    if (number % 2 == 0 || number % 3 == 0) return false; // Check for divisibility by 2 and 3.
    for (int i = 5; i * i <= number; i += 6) { // Check for divisibility by numbers of the form 6k +/- 1.
        if (number % i == 0 || number % (i + 2) == 0) return false;
    }
    return true;
}
```

a natural solution to the equation  $x^2 + y^2 = z^2$  such that  $a \leq x \leq b$  and  $c \leq y \leq d$ .

The program input consists of four natural numbers  $a, b, c, d$  such that  $a \leq b$  and  $c \leq d$ .

The program output should be a line with a natural solution to the equation  $x^2 + y^2 = z^2$  that fulfills  $a \leq x \leq b$  and  $c \leq y \leq d$ .

If there is more than one solution, print the one with the smallest  $x$ .

If there is a tie in  $x$ , print the solution with the smallest  $y$ .

If there are no solutions, print "No solution!".

For example, if the input is: 2 5 4 13

The output should be

$3^2 + 4^2 = 5^2$

and if the input is: 1 1 1 1

The output should be No solution!

Con esta especificación, el bot entrega una solución que es aceptada (AC) por *Jutge.org* pero que no cumple algunos de los requisitos que se exigen en ASIG. En particular, el código de la solución utiliza  $a, b, c, y d$  como identificadores de variables, tal como frecuentemente hacen los estudiantes, puesto que así figuran en la especificación. No obstante, después de sucesivas interacciones con el bot en el que se le indica:

1. Could you please avoid the use of single character identifiers and use meaningful names instead?
2. Please, use a function called from `main()` in your solution
3. Can you make your code compliant with the Google Style Guide for C++?
4. Can you avoid the use of "using namespace std"?
5. Can you write a space on both sides of any binary operator, as the Google Style Guide for C++ requires?
6. Can you finally include Doxygen format header comments in the code?

Se obtiene un código que es aceptado por *Jutge.org* y que cumple los requisitos exigidos al alumnado de ASIG en la evaluación de sus prácticas.

Consideremos a continuación el problema *Primality*. Se trata de determinar si cada uno de los números naturales de una secuencia es o no primo. Al entregar a *ChatGPT* el enunciado del problema en inglés junto con los tests del mismo, tal como figuran en *Jutge.org*, la solución que se obtuvo fue la de fuerza bruta consistente en probar para el número  $N$  todos los divisores en el rango  $[2, N - 1]$ . Esta solución no recibe el veredicto "Accepted" en *Jutge.org* sino que la plataforma indica como veredicto *Execution Error (time limit exceeded)*. Ello se debe a que espera un algoritmo más eficiente para este cómputo.

Esta solución es la que cabría esperar de un estudiante de primer curso de informática o de un programador inexperto. Con frecuencia encontramos estudiantes que, para este problema ofrecen un algoritmo óptimo y ello es una pista para detectar que, posiblemente han hallado la solución en algún foro. Al preguntar la razón por la que no recorre todo el rango de búsqueda cabría esperar una respuesta en la que el programador indique que ha investigado el problema y aprendido sobre el mismo, pero es frecuente una respuesta del tipo "lo he probado y he observado que funcionaba", que obviamente no se considera adecuada en la evaluación de un ejercicio práctico.

En el propio enunciado del problema se indica una pista para un algoritmo de menor complejidad. Si a *ChatGPT* se le indica: *Could you optimize the is\_prime() function for a better performance?* el bot modifica la función, entregando en este caso una versión que sí es aceptada en *Jutge.org*. A pesar de haberlo indicado en el enunciado del problema, la función entregada no cumple con el convenio que establece la Guía de Estilo de Google, pero si se le indica esa circunstancia, el bot corrige el identificador de la función. El código completo de la función es el que se muestra en el Listado 1.

## Listado 2: Declaración preliminar de la clase *Box*

```
class Box {  
public:  
    int length() const { return length_; }  
    int width() const { return width_; }  
    int height() const { return height_; }  
    double weight() const { return weight_; }  
private:  
    int length_;  
    int width_;  
    int height_;  
    double weight_;  
};
```

En el caso de *Primality*, en una sesión diferente con *ChatGPT* se le entregó el enunciado en español y en ese caso, el bot devolvió directamente la solución con la optimización del código, pero la solución no cumplía con el estándar de Estilo en cuanto a la colocación de las llaves de apertura y cierre de bloques en C++.

Particularmente interesante nos ha resultado el caso del experimento realizado con el problema *Increasing Pairs*, consistente en calcular el número de pares de números consecutivos en una secuencia en los que el segundo número del par sea mayor que el primero. Suministrándole a *ChatGPT* la breve descripción del problema en inglés tal cual figura en *Jutge.org*, el bot entrega una solución que, si bien pasa los tests públicos no consigue el veredicto AC del juez porque falla en algún test privado. Hay que tener en cuenta que solo se sabe que el programa falla para algún test (secuencia de números) pero no se sabe más sobre ella. Informando al bot de esta circunstancia, éste responde con una segunda versión de la solución en la que ha resuelto algún caso particular en el que, efectivamente admite que su primera versión fallaría. Esta segunda versión adolece del mismo problema que la anterior: falla en algún test privado de *Jutge.org*, que por lo tanto no la acepta. Se le vuelve a indicar el fallo a *ChatGPT* y finalmente produce una solución que *Jutge.org* valida.

Este modo de trabajo, interactuando con el bot para refinar la solución en sucesivas iteraciones fue el que se utilizó para el experimento que se realizó con un problema consistente en la implementación en C++ de una clase para representar “cajas”. Se trata de un problema de programación que ha figurado como ejercicio de evaluación, en este caso en exámenes escritos, del alumnado de ASIG. Se exponen a continuación en español las sucesivas peticiones que se realizaron a *ChatGPT* en inglés:

1. A continuación (véase el Listado 2) se muestra la definición parcial en C++ de una clase *Box* que representa una caja.
2. Queremos que el código siga las recomendaciones de la Guía de Estilo de Google para C++. Completa la definición de la clase *Box* incluyendo los constructores y métodos que consideres adecuado.

El bot responde incluyendo en la definición de la

clase un par de constructores, métodos mutadores (*setters*) y un par de métodos que permiten calcular volumen y área de la superficie de la caja. *ChatGPT* informa asimismo de que declaración e implementación de la clase deberían ubicarse en sendos ficheros con extensiones `.cpp` y `.h`.

3. Escribe ahora la definición de los métodos que deberían incluirse en el fichero `.cc`.
4. Implementa la sobrecarga del operador de inserción (`<<`).
5. ¿Podrías añadir comentarios en formato *Doxygen* para la función `operator<<?`
6. Implementa la sobrecarga del operador de extracción (`>>`).
7. Implementa la sobrecarga de operadores requerida para comprobar si una caja está antes que otra en un listado ordenado por volumen de mayor a menor.
8. Escribe el código de una función `main()` en la que se instancien 2 objetos de tipo *Box* con volúmenes diferentes y se muestre por pantalla cuál de los dos se encuentra antes en el listado de volumen.
9. Escribe un fichero *Makefile* para compilar el programa usando tres ficheros: `main-box.cc`, `box.cc` y `box.h`. El compilador que queremos usar es `g++`.
10. ¿Qué ficheros deberían incluirse en `main-box.cc` para que el programa compile correctamente?
11. ¿Qué declaraciones debería incluir en el fichero `box.h` para que el programa compile correctamente?
12. Al compilar obtengo el siguiente error:

```
In file included from  
main-box.cc:1: box.h:94:6:  
error: 'ostream' in namespace  
'std' does not name a type  
ChatGPT responde indicando:
```

*The error message you provided indicates that the compiler is unable to find the definition of `std::ostream`. This is likely because the `<iostream>` header file, which defines the `std::ostream` class, has not been included in the `box.h` file. To fix this error, you need to include the `<iostream>` header file at the beginning of `box.h`.*

A cada una de las peticiones anteriores el bot responde entregando sucesivos fragmentos de código con sus correspondientes explicaciones. Una vez corregido el último error señalado, el programa completo con los tres ficheros puede compilarse con `make` obteniéndose un código ejecutable que funciona de acuerdo a la especificación. El listado 3 muestra el contenido del fichero `box.cc` en el que se han eliminado los comentarios en formato *Doxygen* que el bot incluyó antes de cada una de las funciones.

Listado 3: Contenido del fichero `box.cc` generado por *ChatGPT*

```
#include "box.h"
// Constructors
Box::Box() : length_(0), width_(0), height_(0), weight_(0) {}
Box::Box(int length, int width, int height, double weight)
    : length_(length), width_(width), height_(height), weight_(weight) {}
// Accessors
int Box::length() const { return length_; }
int Box::width() const { return width_; }
int Box::height() const { return height_; }
double Box::weight() const { return weight_; }
// Mutators
void Box::set_length(int length) { length_ = length; }
void Box::set_width(int width) { width_ = width; }
void Box::set_height(int height) { height_ = height; }
void Box::set_weight(double weight) { weight_ = weight; }
// Other methods
double Box::Volume() const { return length_ * width_ * height_; }
double Box::SurfaceArea() const { return 2 * (length_ * width_ + width_ * height_ + height_ * length_); }
std::ostream& operator<<(std::ostream& os, const Box& box) {
    os << "Box(" << box.length() << "x" << box.width() << "x" << box.height()
        << ",_" << box.weight() << ")";
    return os;
}

std::istream& operator>>(std::istream& is, Box& box) {
    int length, width, height;
    double weight;
    char x; // to read the 'x' character between dimensions
    char comma; // to read the ',' character between weight and dimensions
    is >> length >> x >> width >> x >> height >> comma >> weight;
    box.set_length(length);
    box.set_width(width);
    box.set_height(height);
    box.set_weight(weight);
    return is;
}

bool operator>(const Box& lhs, const Box& rhs) { return lhs.Volume() > rhs.Volume(); }
```

## 4. Conclusiones

Como muestran los experimentos reflejados en la sección anterior, la capacidad de *ChatGPT* para resolver problemas sencillos de programación como los que se utilizan en los primeros cursos del Grado en Ingeniería Informática es sobresaliente. Podemos afirmar que el asistente es capaz de resolver correctamente la mayoría de los problemas especificados en la plataforma *Jutge.org*, cumpliendo no solo con el requisito de un correcto funcionamiento sino también con los de buenas prácticas y estilo de programación que se exigen en ASIG.

Queda patente que en manos de un programador experimentado, el sistema es -ya en su estado actual- una herramienta muy valiosa para la realización de tareas que pueden entrañar un cierto grado de sofisticación. Interactuando con el sistema es posible refinar en pasos sucesivos una solución dirigiéndola adecuadamente hasta lograr satisfacer unos requisitos predeterminados por el usuario.

Una relación no exhaustiva de preguntas que los experimentos expuestos anteriormente motivan y sobre las cuales debemos reflexionar son las siguientes:

- ¿Suponen los avances en IA conversacional el fi-

nal de la necesidad de la programación de ordenadores como hasta ahora la hemos conocido?

- ¿Debemos ignorar la existencia de los asistentes basados en IA o se necesita por el contrario, incorporarlos a nuestra práctica docente?
- ¿Qué cambios debemos introducir en las prácticas docentes de la enseñanza de la programación?
- ¿En qué momento del itinerario formativo en materia de programación debiera incorporarse el conocimiento y manejo de los asistentes de programación?
- ¿Son útiles los asistentes en los niveles iniciales del aprendizaje de la programación o por el contrario su uso debe aplazarse a niveles posteriores?
- ¿Hay algún peligro en el uso de asistentes que debiéramos tener en cuenta a la hora de exponer al alumnado a estas herramientas?

No es objetivo de este trabajo ofrecer respuestas a estas y otras preguntas que cabe plantear, sino propiciar la discusión y el intercambio de ideas sobre el nuevo escenario que tenemos que afrontar en las aulas de forma inmediata.

Si tal como muchos expertos vaticinan [10], el desarrollo de grandes modelos de lenguajes supondrá en

breve la desaparición de los programadores de ordenadores, la enseñanza de la programación habrá de modificarse para adaptarse a esta nueva realidad. Del mismo modo que en la actualidad un desarrollador de software en general desconoce los detalles del hardware en el que se ejecutan sus programas, es posible que en el futuro un ingeniero de software no tenga que conocer los detalles del código que interviene en un sistema complejo. Habrá en este caso que desplazar progresivamente el énfasis en los contenidos de programación hacia conocimientos relacionados con ingeniería y arquitectura del software.

Hurtar al alumnado el uso de los asistentes basados en IA generativa en sus prácticas de programación [1], además de ser un esfuerzo vano, creemos que sería un enfoque profundamente erróneo. Se trata de herramientas que ya forman parte del bagaje del que un profesional dispone para desarrollar su trabajo. Del mismo modo que carece a nuestro juicio de sentido evaluar las capacidades de programación de un estudiante mediante exámenes tradicionales con bolígrafo y papel, tampoco tiene sentido prohibirles el uso de asistentes que en el futuro tendrán que conocer y utilizar. En lugar de prohibir a los estudiantes el uso de estos sistemas, creemos que lo que ha de hacerse es ayudar al profesorado y al alumnado a incorporar estas nuevas herramientas como elementos de apoyo al aprendizaje.

La situación que ahora enfrentamos no es nueva: la aparición de Wikipedia o foros de discusión como *StackOverflow* en los que se ofrecen soluciones a multitud de problemas de diferente tipo supuso en su momento una ampliación del conjunto de herramientas disponibles a la hora de programar y de aprender a programar. Es lo mismo que ocurre hoy en día, pero agravado si cabe, porque las capacidades que en la actualidad muestran los nuevos asistentes son realmente impresionantes y es difícil aventurar lo que pueden llegar a ofrecer en un corto o medio plazo. De hecho, son las prestaciones que ya ofrecen estos asistentes lo que más ha llamado la atención a especialistas en IA: se esperaba que estos adelantos se fueran a producir pero ha causado cierta sorpresa el haberlos logrado en la actualidad.

Si bien la llegada de los asistentes basados en IA a las aulas no implicará que el profesorado deje de ser necesario, sí es cierto que estas tecnologías van a impactar en la praxis docente y hemos de adaptar nuestras metodologías para incorporar estos cambios. Integrar estos asistentes en las metodologías docentes e instruir a profesorado y alumnado acerca de su funcionamiento

y de sus posibles usos será clave para lograr una utilización correcta de estos sistemas.

## Referencias

- [1] Caitlin Cassidy. Australian universities to return to 'pen and paper' exams after students caught using AI to write essays. *The Guardian*, enero 2023. <https://www.theguardian.com/>.
- [2] Davide Castelvechi. Are ChatGPT and AlphaCode going to replace programmers? *Nature (London)*, 12 2022.
- [3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, y Greg Brockman. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [4] Jordi Pérez Colomé. "funciona muy bien, pero no es magia": así es ChatGPT, la nueva inteligencia artificial que supera límites. *El País*, diciembre 2022. <https://elpais.com/>.
- [5] Tiffany H. Kung et al. Performance of ChatGPT on USMLE: Potential for AI-Assisted Medical Education Using Large Language Models. En *NPJ Digit. Med.*, 2022.
- [6] Luciano Floridi y Massimo Chiriatti. GPT-3: Its nature, scope, limits, and consequences. *Minds Mach.*, 30(4):681–694, dec 2020.
- [7] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, y Julian Schrittwieser et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.
- [8] Nhan Ton Nguyen y Sarah Nadi. An Empirical Evaluation of GitHub Copilot's Code Suggestions. *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, pp. 1–5, 2022.
- [9] Jordi Petit, Josep Carmona, Jordi Cortadella, Jordi Duch, Giménez Omer, Anaga Mani, Jan Mas, y Enric Rodríguez-Carbonell. Jutge.org: Characteristics and experiences. *IEEE Transactions on Learning Technologies*, 11(3):321–333, 2018.
- [10] Matt Welsh. The end of programming. *Communications of the ACM*, 66(1):34–35, 2023.
- [11] Xingxing Zhang, Jinchao Zhang, y Maxine Eskenazi. ChatGPT: A task-oriented dialogue system based on pre-trained language model. *arXiv preprint arXiv:2012.16641*, 2020.