Programación en la escuela secundaria

Autor: Gabriel Sánchez

Año: 2023

Índice

Resumen

Enseñar programación puede ser difícil. El código fuente es la forma en que le comunicamos a la computadora lo que queremos que haga. Este texto ofrece una perspectiva personal sobre cómo enseñar programación en la escuela secundaria. El autor explorará las ventajas y desventajas de diferentes enfoques, lenguajes de programación, marcos, software y otras tecnologías utilizadas en la educación. También explora otros temas como el control de versiones y la legibilidad del código.

Habla sobre su experiencia y cómo influye en su enfoque pedagógico y enfatiza la necesidad de considerar las habilidades de inglés de los estudiantes, ya que la sintaxis de muchos lenguajes de programación se basa en esa lengua y es una buena práctica usarla para escribir cosas como nombres de variables. Otra cosa que recomienda es usar Linux para codificar y enseñar algoritmos en lugar de un lenguaje específico y seguir una receta. También profundiza en algunas buenas prácticas, como no depender de comentarios en el código. Gabriel habla sobre los motores gráficos utilizados para crear videojuegos y cómo pueden hacer que el aprendizaje sea divertido para los estudiantes.

En definitiva, el autor dice que es importante proporcionar una introducción a la programación en educación secundaria, adaptándola al contexto e intereses de los estudiantes. La monografía ofrece una visión personal sobre cómo abordar esta tarea y promover el aprendizaje significativo de la informática en el aula.

Palabras clave: Programación - Enseñanza - Computadoras - Código - Inglés

Abstract

Teaching programming can be difficult. Source code is the way we communicate to the computer what we want it to do. This text offers a personal perspective about how to teach programming in high school. The author will explore advantages and disadvantages of different approaches, programming languages, frameworks, software and other technologies used in education. Also he explores other topics like version control and code readability.

He talks about his experience and how it influences his pedagogical approach and emphasizes the need to consider students' English skills since many programming languages syntax are based on that tongue and it is a good practice to use it to write things like variable names.

Another thing he recommends is using Linux for coding and teaching algorithms instead of a specific language and following a recipe. He also dives into some good practices like not relying on comments in code. Gabriel talks about graphics engines used to make video games and how they can make learning fun to students.

Ultimately, the author says that it is important to provide an introduction to programming in secondary education, tailoring it to the context and interests of the students. The monograph offers a personal insight into how to approach this task and promote meaningful learning of computer science in the classroom.

Keywords: Programming - Teaching - Computers - Code - English

Introducción

En este texto voy a explorar desde mi experiencia personal en programación cuáles son las ventajas y desventajas de enseñar a programar de cierta manera en varios lenguajes de programación y tecnologías.

También voy a hablar de diversos temas, como las buenas prácticas de programación. Al escribir la monografía yo menciono términos técnicos que van a requerir buscar en internet qué significan porque lo escribo desde mi experiencia previa. Escribo esta monografía como trabajo final del seminario de Trabajo y Rol Docente de la Certificación Pedagógica.

En esta carrera hay quienes van a enseñar en nivel medio, otros en nivel primario y otros en ambos. En mi caso, voy a enseñar en nivel medio por lo que pensé escribir para la secundaria.

Otra de las cosas que me motiva a escribir este texto es mi pasión por los videojuegos, la que a su vez me llevó a querer crear los mios. Considero que hay muchos adolescentes a los que también les puede interesar y pueden aprender mejor jugando que con lógica de negocio.

Durante la escritura del desarrollo de este escrito también busqué ponerme en el lugar de alguien que no sabe programar y traté de recordar cómo me sentía cuando me estaban enseñando cosas que no sabía.

Otra de las cosas que traté de hacer incluso para investigar fue intentar enseñarle JavaScript a mi papá. Escribir este texto no hubiera sido posible sin el trabajo de los desarrolladores de software libre de Linux, Fedora, Vim, Terminator, Neovim y Git. Esto es debido a que usé gratis sus programas para escribir el presente trabajo.

Desarrollo

¿Para quién está pensada ésta monografía? Para gente que sepa de programación y esté interesado en enseñar. Considero que enseñar a programar en el nivel medio muchas veces está mal enfocado.

En mi caso me ha pasado que nos enseñaron cosas como HTML (al cual incorrectamente definían como un lenguaje de programación) o robótica. También es común que se usen otras cosas como Scratch y Arduino. Otras veces se les enseña incluso a usar programas de animación como Alice. Yo creo que lo primero que se tiene que enseñar es a

usar control de versiones, especialmente Git. Después de eso se tiene que elegir un lenguaje de programación.

En la carrera de Analista de Sistemas, lo primero que me mostraron fue cómo usar pseudocódigo, en parte porque la mayoría de los lenguajes de programación tienen sintaxis en inglés. Hay que evaluar el nivel de inglés de los alumnos porque no todos pueden entender inmediatamente lo que van a escribir.

Usar HTML puro tiene sus ventajas porque se pueden generar páginas que cargan más rápido y pesan menos que si se usan cosas como React, pero al momento de hacer cosas que no sean estáticas como centrar el texto o usar código de JavaScript puede ser muy difícil hacer páginas complejas.

Hacer programas de consola en los cuales se imprimen resultados puede ser bastante abstracto para los alumnos. Programar aplicaciones web con frameworks como React como primera experiencia con la programación puede ser muy difícil. Usar motores gráficos de juego puede ser abrumador al principio porque son complejos. Creo que usar librerías o consolas de fantasía como TIC-80 puede ser una buena idea. También creo que es importante poder elegir un sistema operativo adecuado para programar.

En mi experiencia personal es ideal poder usar linux, pero las computadoras que hay en las escuelas de la Ciudad Autónoma de Buenos Aires, hoy en día todas tienen únicamente Windows. Una alternativa sería que se pudieran instalar máquinas virtuales con linux en esas computadoras pero en muchos casos no se permite instalar nada en esas laptops. Otra solución sería usar Linux en un USB, pero esto tiene sus limitaciones, requeriría conseguir los usb y el bootloader de la máquina debería estar desbloqueado. En los únicos casos en donde windows puede ser una mejor opción para programar es si se usa algún programa que no está disponible de forma nativa para Linux. Las primeras laptops que habían repartido en las escuelas en Buenos Aires tenían una partición de Debian o Huayra, pero ahora solo tienen el sistema operativo de Microsoft.

Es importante que cuando se le enseña a programar algo a los alumnos no se les enseñe paso a paso cómo hacer un juego en específico, sino que hay que enseñarle algoritmos para que puedan programar prácticamente lo que quieran. “En el contexto informático, un algoritmo es una secuencia de pasos computacionales, es decir, una secuencia de pasos de entre un repertorio fijo, que son aquellos que puede hacer una computadora” (Schapachnik, p. 55, 2022).

Además quiero destacar que hacer un juego en dos dimensiones es mucho más fácil que hacer juegos en tres dimensiones o con realidad aumentada. En mi tiempo como

programador he probado mucho software diferente para hacer juegos o programas. Pygame, Phaser.js, Paper.js, Godot, Unity, Unreal Engine, TIC-80, entre otras. Hay algunas de más bajo nivel escritas en Lua, C, C++, C# o Rust que no pude hacer andar o son muy difíciles. Todas tienen sus ventajas y desventajas. Algunas son de código abierto y otras son propietarias. En el caso de TIC-80 es de código abierto, pero cuenta con binarios de pago para apoyar a los desarrolladores, los cuales pueden ser compilados por uno mismo para ahorrarse pagar 10 dólares. Una de las ventajas de TIC-80 que tiene es que es todo en uno. Tiene editor de código, de sprites o imágenes, de celdas, de efectos de sonido y de música. Creo que las limitaciones técnicas llevan a que en la mayoría de los casos se estimule la creatividad y este programa simula ser una computadora antigua.

Otra cosa que sirve para aprender a programar es aprender a leer el código de los demás. Una página muy útil para eso es GitHub, pero también existen otras como GitLab. Esta página también les permite varias cosas como trabajar en proyectos colaborativamente, hacer control de versiones con git, subir una página web estática, lanzar versiones en archivos binarios para varias plataformas. Yo incluso subo mis archivos del profesorado a GitHub para tenerlos en varias computadoras y no perder mis apuntes.

El objetivo del trabajo del docente es que los alumnos aprendan, no solo enseñar. No es cuestión de enseñarle a la pared cómo hacer algo, los alumnos lo tienen que entender. Programar no es solo cuestión de escribir código, de ser así todos los juegos serían polígonos de colores. También hay que diseñar los sprites, los cuales son imágenes que se usan en el juego para representar personajes, fondos, objetos, etc. Algo que es opcional es el sonido ya que en muchos casos los usuarios deciden apagarlo. Los primeros juegos que hice eran totalmente mudos porque era difícil para mí ponerle música.

En muchos motores de juego es necesario usar programas externos para crear el sonido, pero como mencioné antes en las consolas de fantasía como TIC-80 ya tienen creadores de música. La desventaja de esto es que si se quiere poner una canción ya existente se tiene que programar a mano toda la partitura de la canción, pero la ventaja es que si nos gusta la música retro de videojuegos como el chiptune podemos crear con este programa una versión de una canción que todavía no fue adaptada.

Escuché a bastante gente hablar sobre Minecraft Education Edition y como se usa en la escuela. Los juegos de sandbox tienen bastante potencial para hacer que a los alumnos les interese la informática, pero en mi opinión solo pueden en la mayoría de los casos hacer que practiquen diseño de niveles. Si lo que se quiere es que ellos tengan que programar usando un juego ya existente entonces lo que se podría hacer es que hagan un mod de un juego.

Para poder escribir el código fuente de los programas es necesario usar un editor de texto. Hay varios programas que tienen un editor de texto integrado, pero yo recomiendo usar Vim porque es el que usé para empezar a escribir esta monografía. Este editor se ejecuta en el emulador de terminal pero también tiene una versión independiente. Considero que enseñar a usar este editor de texto a los alumnos puede ser muy útil tanto para programar como para escribir apuntes o textos. La particularidad que lo diferencia de los demás editores es que es muy ligero y tiene modos que permiten editar más fácilmente. Lo primero que se le tiene que enseñar a alguien es como salir de este programa, lo cual se hace usando :q y luego enter. Cuando se apreta Esc se entra en el modo normal, desde el cual se pueden acceder a otros modos y usar comandos. Casi todas las letras del teclado tienen una función. Para empezar a escribir al entrar en el modo insertar se puede usar la letra I o la A.

Cuando le pregunté a un profesor de cívica e inglés de como haría una microclase de informática para Taller 1 lo que me respondió es enseñar a usar PowerPoint. En mi opinión no se si es la mejor opción, yo preferiría darles una introducción a javascript.

"Enseñar no existe sin el aprender" (Freire, p. 45, 1993). Yo puedo dar una exposición sobre un tema muy avanzado de programación y los alumnos pueden no entender lo que estoy diciendo. Para que aprendan primero tengo que ponerme en el lugar de ellos y pensar en cómo era yo cuando no sabía estos temas o mientras los estaba aprendiendo.

"El acto de estudiar siempre implica leer" (Freire, p. 47, 1993). Para aprender a escribir código, primero tenemos que leer el código de otros, ya sea en documentación, foros, clases, repositorios online, etc. Es importante escribir código que sea legible.

Hay gente, como mi profesor de Taller de Programación, que prefiere hacer comentarios en el código, pero recientemente vi en un video que no conviene usar comentarios. Existen varias razones. La primera es que los comentarios son ignorados por el compilador o el intérprete, por lo que cuando se modifica el código los comentarios pueden quedar desactualizados. La segunda es que si se usan buenas prácticas, como no abreviar los nombres de variables, clases y funciones, no es necesario poner comentarios. Una mala práctica es comentar código que no queremos que se ejecute. Lo recomendable es directamente hacer un “commit” y borrar ese código, porque cuando alguien vea ese código en el futuro no va a querer borrarlo por pensar que es importante.

No podemos esperar que el escritor haga su tarea y la del lector (Freire, p. 53, 1993). Escribiendo esta monografía constantemente estuve pensando en que por ahí escribía demasiados ejemplos o explicaba de más qué significaba cada término. Para poder leer este

texto va a ser necesario tener conocimiento previo de informática o buscar en internet a que me refiero.

Es recomendable escribir por lo menos 3 veces por semana (Freire, p. 57, 1993). Hay que tener en cuenta que esta carta fue escrita en el siglo XX. Hoy en día, por la invención de la mensajería instantánea y las redes sociales, escribimos mucho más que en el pasado. Considero que la formación profesional que tiene una persona influye en como escribe cada uno. En mi caso como soy programador prefiero escribir una oración por renglón y en lugar de usar el teclado con Ñ y tildes uso el de inglés estándar porque es el que se suele usar para programar. Lo que sí es cierto y se menciona un poco antes en esa carta es que a muchas personas les cuesta escribir y redactar correctamente en tesis o trabajos prácticos. Esto podría aplicarse también a escribir código. Aunque no tengamos un trabajo o estemos estudiando formalmente programación, es importante que sigamos practicando programar, de manera autodidacta. Lo bueno es que si nos gusta programar podemos desarrollar el proyecto que nosotros queramos.

En mi primer día de clases de la carrera de Analista de Sistemas me acuerdo que el profesor preguntó si teníamos experiencia previa con la programación y yo le comenté que había aprendido algo de HTML. Él me dijo que técnicamente no era un lenguaje de programación.

Otra cosa que estuve pensando es si realmente se le debería enseñar programación a los chicos en el secundario, en parte porque puede ser que no les interese del todo el tema. “La educación general no tiene como objetivo formar a los profesionales que el sector informático requiere; pero la escuela si tiene el rol fundamental que cumplir en el momento de despertar estas vocaciones e incentivar la participación” (Schapachnik, p. 55, 2022).

Yo creo que depende, dado que si es una escuela orientada a ciencias sociales como el Juan B. Justo al cual yo asistí, no va a tener la misma cantidad de horas disponibles que en una escuela técnica orientada a computación. De todas formas, como dice en la cita, deberían darles aunque sea una introducción al tema.

Una de las incertidumbres que tuve al escribir esta monografía es que no estoy seguro si estoy capacitado para enseñar Tecnología en nivel medio porque mi formación es en su mayoría sobre programación. Yo no sé mucho sobre cómo enseñar Office tampoco, ni hablar sobre las cosas que se enseñan normalmente en el primer ciclo del secundario en esta materia. Por eso creo que quizás lo mejor para mi sería enseñar en un colegio técnico o buscar específicamente hacer concurso para enseñar informática o tecnologías de la información.

También tuve la inquietud de que por ahí lo que yo quiera y sepa enseñar no esté dentro del plan de estudios.

No todos los programadores tienen un mismo primer lenguaje de programación, pero en mi caso fue Java y mis profesores lo recomendaban porque al tener tipos de datos es más fácil aprender primero a usarlos que tener como primer lenguaje a uno como python o javascript los cuales no tienen tipos. La idea también era que al ser un lenguaje un poco más difícil y estricto entonces podríamos aprender más rápidamente y los otros lenguajes después nos parecen más fáciles.

Conclusión

En esta monografía se exploró con profundidad la enseñanza de la programación en el nivel medio, desde una visión personal sobre este tema. A lo largo de esta se mostraron distintas posibilidades y qué implican. Lo que quedaría es llevarlas a cabo en un ambiente real en el que se enseñe a personas que no tengan experiencia previa para poder experimentar si lo que en teoría para alguien que ya sabe realmente pueda ser aceptado y entendido.

En la materia del Taller 1 de la Certificación Pedagógica yo dí dos microclases. La primera era sobre las aplicaciones con herramientas de texto a voz como el Traductor de Google y la segunda era sobre cómo usar el editor de texto Vim que mencioné previamente en la introducción.

En esa segunda instancia lo que hice fue sentarme mirando hacia el público usando la computadora y la conecté a una pantalla grande. Le pedí a los estudiantes que instalen una aplicación de Vim para el celular para que usen mientras yo doy la clase. Después lo que hice fue explicarles que hacía cada comando. Finalmente les pasé el link para que respondan preguntas usando Mentimeter sobre cómo usar Vim.

En esa microclase yo estuve en grupo con una compañera bibliotecaria pero uno de los inconvenientes que tuvimos fue tratar de juntar nuestra incumbencias. El tema que elegimos fue el software de texto a voz y las leyendas. Por lo que dijo el profesor estos temas no tenían nada que ver y parecía como que estuviéramos dando dos clases distintas.

Hoy en día tenemos muchas herramientas para generar código que los primeros programadores no tenían, como Chat GPT. Esto no significa que estos sean reemplazables, al contrario. Para poder usarlas es necesario saber como instalar y ejecutar lo que genera la inteligencia artificial. No cualquiera puede decirle a la computadora que haga algo sin entender qué es lo que quiere hacer y cómo funciona. También está el tema del bajo nivel y el

alto nivel, el cual cuanto más bajo cerca está del funcionamiento interno de la máquina. Lo más bajo sería por ejemplo escribir en binario directamente a la computadora y lo más alto es escribir en un idioma humano que queremos que haga el ordenador.

Bibliografía

Freire, Paulo; Cartas a quien pretende enseñar - 2° ed. 5° reimp.- Buenos Aires: Siglo Veintiuno Editores, 2010. Título original: Professora sim; tia não: cartas a quem ousa ensinar, 1993.

Schapachnik, Fernando; Ciencias de la Computación en la Escuela - 1° ed.- Buenos Aires: Siglo XXI Editores Argentina, 2022.