Niños aprendiendo a programar: el nuevo lenguaje

Alumno: Fernando Abellán Contreras

Máster en Educación Secundaria

Especialidad: Tecnología e Informática

Profesor UNIR: Álvaro Nolla de Celis

1. Índice

2. Resumen

El presente trabajo trata de estudiar y analizar cómo influye en los niños la enseñanza de la programación en edades tempranas.

No se trata de hacer un análisis centrado en hechos concretos si no en analizar y asimilar cuál es la situación actual en torno a esta idea.

En este trabajo trataremos de asentar unas bases teóricas mínimas para que cualquiera pueda asimilar el lenguaje común que se utiliza en los entornos de desarrollo. Por ello se dedica un punto a definir y describir los conceptos básicos. No es necesario ningún conocimiento previo.

También se trata el tema temporal sin ahondar en demasía en detalles históricos que poco pudieran interesar, en una primera aproximación, a un lector poco cercano a este mundo. Se hace una pequeña revisión de la historia de la informática que transmita la importancia y peso que esta joven disciplina ha tenido y tendrá desde su nacimiento.

Después una búsqueda entre las fuentes disponibles, se seleccionaron aquellos análisis que, a juicio del autor de este trabajo, eran los más acertados. En este trabajo se tratan las principales ideas difundidas por la profesora Jeannette M. Wing y la asociación “Partnership for 21st Century Skills” para argumentar habilidades y desarrollos fomentados por este tipo de enseñanzas.

En el presente trabajo, también se citan aquellos casos más significativos en los que la idea de implantar estos tipos de enseñanza están teniendo mayor repercusión a nivel internacional. Además, en menor medida, también se ha estudiado y aborda la situación España donde, tal y como veremos, es más bien pesimista.

Por último se han seleccionado ciertas herramientas, lenguajes o recursos didácticos acompañados de una pequeña introducción para acercarlos al docente y difundir su conocimiento

3. Introducción

3.1. Motivación

Durante toda mi vida he tenido un ordenador cerca. Mi padre, además de maestro, es un gran aficionado a todos estos "aparatos".

Que mi madre también sea maestra sólo sirve para reafirmar que me influyeron en la fascinación y el respeto que hoy en día tengo por la enseñanza.

Esos dos factores, la tecnología y el aprendizaje, han sido muy influyentes en la persona que soy ahora.

No sé si creer que nacemos con cierta predilección por algunas cosas o, por el contrario, nos amoldamos a lo que nos rodea. Quizá, si no llego a tener cerca aquel libro de programación en Basic y mi padre no hubiera comprado aquellos ordenadores, tal vez, hoy en día, no llevaría más de catorce años programando.

Pero resulta que sí. Soy programador y vivo de ello. Creo que lo soy desde antes de que empezara a programar. Creo que todos, en cierta manera, lo somos.

A las personas nos gusta hacer las cosas a nuestra manera. Seguir un determinado orden al meter las maletas en el coche, dejar recados en la nevera, tomar notas, organizar nuestra biblioteca,... Todas estas cosas implican una necesidad de controlar una pequeña parcela del mundo. Nos suele gustar sentir que controlamos lo que tenemos a nuestro alcance. Nos gusta programar el mundo.

Creo fehacientemente que aprender a programar a edades tempranas puede "amueblar" el cerebro a pensar de forma diferente. Puede ayudar a enfocar los problemas rutinarios de otra forma y, además, puede desarrollar otras habilidades o disciplinas tales como métodos de estudio, resolución de problemas, trabajo en equipo, gestión de recursos,...

Nos encontramos en un país donde, en la mayoría de casos, mi profesión como programador se encuentra prostituida y muy subestimada. Es una minoría la que, en el entorno empresarial, valora como se merece la difícil labor de un programador. El desarrollo y éxito profesional en mi sector siempre conlleva, muy a mi pesar, un alejamiento paulatino de las labores técnicas en favor de otras tareas de gestión. Hoy en día, en España, es muy difícil encontrar un programador jubilado. Partiendo de que esto es un problema cultural muy difícilmente extirpable, me gustaría analizar cómo se encuentra la educación de la disciplina computacional en nuestro país y pronosticar que un aumento de su presencia en la enseñanza de nuestros niños, en

términos generales, podría influir en un posible cambio de la percepción negativa que se tiene de la programación.

En un mundo globalizado, movido por un dinamismo salvaje, guiado por tendencias que duran minutos, y gestionado a través de dispositivos móviles, tenemos dos opciones: depender de las máquinas o hablar con ellas.

Enseñar a los niños a programar es, en resumen, enseñarles “El nuevo lenguaje”.

3.2. Objetivos

3.2.1. Objetivo general

El objetivo principal de este trabajo es profundizar en la idea de que enseñar a los niños a programar desde edades tempranas supone una ventaja palpable para su desarrollo personal y profesional. Se tratarán de analizar en profundidad diversas fuentes, y obtener una visión general de este aspecto para concluir con una idea formada y justificada.

3.2.2. Objetivos específicos

En relación al tema central del trabajo, se pretende:

- Realizar una pequeña redacción de un marco teórico en el que se moverá este trabajo. Un punto que se tratará es el de la necesidad detectada de que los profesores se formen en este ámbito para que los niños reciban la enseñanza adecuada. Este trabajo, ayudado de este marco teórico, quiere ser un apoyo y un punto de partida para aquellos profesores interesados en iniciarse en los conceptos básicos del mundo de la programación.
- Analizar el contexto histórico en el que se han movido la enseñanzas relacionadas con las ciencias de la computación. Se pondrán ejemplos de herramientas, éxitos y logros acontecidos a lo largo de la historia.
- Se profundizará en los efectos que supone en los niños el aprendizaje temprano de la programación. Se analizarán los puntos de mejora y las habilidades adquiridas.
- Se analizará la situación actual tanto a nivel nacional como internacional. Se prestará especial atención al Reino Unido donde actualmente se está ejerciendo, por parte de diversos medios, una presión especial a favor de la inclusión de estas enseñanzas.
- Se investigará y obtendrán casos reales para ejemplificar cómo está impactando esta enseñanza en nuestra sociedad actual.
- Se hará una labor de análisis de distintas herramientas que sirvan para la enseñanza de la programación en niños.

3.3. Metodología

Para hacerme una idea general sobre los efectos y contexto creado en torno a la idea de niños aprendiendo a programar he estudiado y leído artículos periodísticos, trabajos académicos, libros de opinión, libros teóricos y, sobre todo, recursos digitales creados por usuarios particulares con determinados intereses.

Parto de una serie de corrientes detectadas en determinados medios de comunicación, por lo que he profundizado en cada idea que encuentre. Después de asimilar he utilizado todos los conceptos encontrados para vertebrar el trabajo en ciertas ideas que le den sentido como conjunto.

También estoy interesado, desde hace algo más de un año, en un concepto denominado "pensamiento computacional". He centrado parte de mi estudio bibliográfico en este concepto.

Partiendo de estas ideas he leído y contrastado muchos recursos, como ya he dicho en su mayoría digitales, que he sintetizado y ordenado para que terminen alineados con los objetivos iniciales que me marqué.

Por último, además de encontrar ideas, he identificado las herramientas más interesantes y he planteado, previa asimilación, un ejemplo básico que sirva de aperitivo para cualquier interesado.

4. Aportaciones del trabajo

Este trabajo tiene como aportación establecer un punto de partida común a una serie de conceptos que no son muy comunes en nuestro entorno. Así, trataremos de asentar una pequeña base de conceptos y términos que sirvan para introducir a cualquier lector en el mundo de la programación y su enseñanza a los niños.

Lo ideal es que este primer paso sea el primero y más fácil para muchos, de forma que cada uno pueda seguir su camino enfocado de acuerdo a sus necesidades.

Como segunda aportación pretendo recopilar determinados recursos para que alumnos y profesores puedan localizar fácilmente determinadas ideas. Así, es importante saber de dónde viene la informática, qué movimientos hay hoy en día, y hacia dónde vamos. También es importante conocer distintas herramientas y tendencias que se están dando en el momento que vivimos.

5. La programación y los niños

5.1. Marco teórico

En este apartado trataremos de responder, de la forma más simple posible, preguntas tan elementales como ¿qué es un programa informático? o ¿qué entendemos por programación?.

Asimismo describiremos los tipos e lenguajes que entiende un ordenador con intención de ubicar un lenguaje de alto nivel y, de este modo, entender su utilidad.

Para acabar haremos un repaso de los elementos básicos de un programa informático desde el punto de vista del desarrollador para, finalmente, definir un pequeño ejemplo a muy alto nivel para mejorar la comprensión de los conceptos aquí tratados.

Partiremos de la idea de que los lectores de este trabajo serán personas interesadas en el tema con perfiles muy diferentes, aunque trataremos de que este capítulo abarque unos pocos conceptos básicos que supongo mínimos para la comprensión del resto del documento desde una perspectiva correcta.

Se trata de una base teórica muy simple que desarrollaremos en función de los conceptos más sencillos del libro “Fundamentos de Programación” (Cerrada Somolinos, José Antonio y Collado Machuca, Manuel E., 2010).

5.1.1. ¿Qué es un algoritmo?

Entendemos por algoritmo a una serie de pasos u órdenes que se definen para lleva a cabo una tarea. Este concepto, de forma aislada, no aplica únicamente a un entorno computacional. En la vida cotidiana se emplean algoritmos para muchas tareas que implican la resolución de problemas, como por ejemplo: instrucciones en un manual para sintonizar los canales de TV; acciones descritas en una receta para cocinar un bizcocho; los pasos a seguir para el cálculo de una raíz cuadrada de forma manual.

5.1.2. ¿Qué es un lenguaje de programación?

Simplificando al máximo la definición, entendemos por lenguaje de programación a un lenguaje creado para expresar acciones que pueden realizar los ordenadores.

Hay varios tipos y, para ubicarnos, los clasificaremos basándonos en su nivel de concreción y haremos una breve descripción de cada uno de ellos.

5.1.2.1. Lenguaje máquina

Este es el único lenguaje que, realmente entienden los ordenadores.

La unidad de información en un sistema informático es el bit: es una estructura mínima que almacena un uno o un cero.

Cualquier dato u orden que almacena un ordenador se representa, en última instancia, utilizando una secuencia de bits. Esto quiere decir que cualquier instrucción que el ordenador ejecuta ha sido previamente traducida a este lenguaje para que las estructuras de proceso internas realicen las acciones oportunas.

Como es fácil prever, este lenguaje es muy difícil de utilizar para un ser humano. Sería imposible recordar todas las instrucciones que un ordenador entiende y comunicárselas utilizando únicamente unos y ceros. Es por ello que se definen lenguajes de más alto nivel.

5.1.2.2. Ensamblador

Se trata de un lenguaje más comprensible que facilita la lectura y escritura de instrucciones a un ordenador.

En lugar de utilizar secuencias de bits, permite la utilización de instrucciones más legibles para realizar determinadas acciones.

El lenguaje ensamblador es dependiente del tipo de ordenador que las ejecute. Por ejemplo, una instrucción puede ser válida para un ordenador con procesador Intel pero no para uno que utilice procesadores *AMD*.

De cualquier forma y, pese a la ventaja frente al lenguaje máquina, sigue siendo un lenguaje poco ameno y de muy bajo nivel.

5.1.2.3. Lenguajes de alto nivel

Es un recubrimiento para hacer más amigable la definición de acciones en un ordenador.

Estos son los lenguajes de los que, posiblemente, la mayoría de personas han oído hablar: *C, C++, Java,…*

La principal ventaja es que se trata de lenguajes que son legibles. Se basan, casi todos, en la lengua inglesa. Esto significa que, en la mayoría de los casos, leer una línea de algo escrito con ellos puede servir para intuir qué es lo que se pretende hacer.

Otra ventaja es que son independientes, también en la mayoría de los casos, de las características de la máquina que los está utilizando.

La única pega es que son más ineficientes. Simplificando mucho el proceso hay que entender que, para que el ordenador entienda lo que queremos que haga, este lenguaje debe traducirse, primero, a lenguaje ensamblador y, finalmente, a lenguaje máquina. Dependiendo de qué estemos utilizando este proceso puede ser más o menos lento.

De cualquier forma, dado que nuestro objetivo final es el desarrollo del razonamiento computacional, no debemos centrarnos en el rendimiento si no en la utilización de lenguajes que propicien la captación del interés del programador (en este caso el niño) y su desarrollo posterior.

5.1.2.4. Pseudocódigo

No es un tipo de lenguaje equiparable a los tres anteriores, pero es necesario su conocimiento para la expresión natural de algoritmos.

Se trata de un paso más en esta escala por hacer legible lo que, finalmente, debe entender la máquina.

Entendemos por pseudocódigo a una forma de describir un algoritmo utilizando una sintaxis cercana a un lenguaje de programación, pero destinada a la comprensión humana.

5.1.3. ¿Qué es un programa de ordenador?

Un programa informático es un conjunto de instrucciones que le dice al ordenador qué debe hacer para llevar a cabo una tarea.

Estas instrucciones son el resultado del diseño de un algoritmo que resuelve el problema en cuestión, y de su implementación utilizando un lenguaje de programación específico.

5.1.4. Elementos básicos en los lenguajes de programación

Para una fácil asimilación de los conceptos aquí tratados, simplificaremos al máximo los conceptos, reduciendo con ello las posibles faltas de comprensión o el rechazo de cualquier lector no iniciado en este mundo.

He decidido inspirarme en la programación estructurada como punto de partida para ayudar a los no iniciados en la comprensión de los conceptos aquí vertidos. Sabido es que esta técnica de programación data de unas necesidades y contexto que distan bastante de los que nos encontramos hoy en día. De cualquier

forma, estimo como suficientes los esfuerzos que la asimilación de esta técnica desencadenará en las mentes de los lectores menos habituados.

Los ejemplos que se escriban en este trabajo, de aquí en adelante, se codificarán utilizando pseudocódigo.

5.1.4.1. Variables

Las variables es lo que utilizan los programas de ordenador para almacenar información durante la ejecución de un programa. El valor de estas puede cambiar durante este tiempo.

Las variables pueden ser de distintos tipos, dependiendo del tipo de información que pretendan almacenar. Los tipos más comunes son: lógico (almacenan dos estados que, generalmente, son verdadero o falso); entero (almacenan un número sin decimales); coma flotante (almacenan un número real, con decimales); carácter; cadena (almacenan una secuencia de caracteres).

5.1.4.2. Asignaciones

Se llama así al proceso que se lleva a cabo cuando queremos cambiar el valor de una variable. Ejemplo:

```
texto = "Esto es el contenido de una cadena de texto";
```

5.1.4.3. Estructuras condicionales

En cualquier programa informático será necesario tomar decisiones en función de la información que tengamos. Las estructuras condicionales nos permiten decirle al ordenador qué debe hacer en función de lo que sabe.

El siguiente fragmento de código muestra un mensaje por pantalla en función del valor de un número:

```
numero = 4
Si numero mayor que 0 Entonces
      mostrar por pantalla "El número es positivo"
Si no
      mostrar por pantalla "El número es negativo"
Fin si
```

5.1.4.4. Estructuras repetitivas (bucles)

Este tipo de estructuras sirven para repetir la ejecución de determinadas instrucciones un número determinado de veces.

El siguiente fragmento cuenta hasta diez repitiendo la misma instrucción de asignación esas mismas veces:

```
numero = 1
Mientras numero <= 10 Hacer
      mostrar por pantalla numero
      numero = numero + 1
Fin Mientras
```

5.1.5. Ejemplo: problema y solución

Imaginemos que queremos desarrollar un algoritmo que recibe dos números (obviaremos los mecanismos que posibilitan esta parte) y debe decidir cuál de ellos es el mayor.

El programa en cuestión sería de la siguiente forma:

```
numero1 = ...
numero2 = ...
...
Si numero1 mayor que numero2 Entonces
      mostrar "El mayor de los dos numeros es: ", numero1
Si no Si numero2 mayor que numero1 Entonces
      mostrar "El mayor de los dos numeros es: ", numero2
Si no
      mostrar "Los dos números son iguales"
Fin Si
```

5.2. Antecedentes e historia

5.2.1. Origen de los ordenadores y lenguajes de programación

El ordenador no es fácilmente ubicable en la historia como cualquier otro invento. La creación del mismo no se debe únicamente a una persona si no a un cúmulo de circunstancias y avances que, en algún momento, confluyeron hasta unas mismas necesidades que se terminaron materializando en lo que conocemos hoy en día. Obviamente esta evolución fue también fruto de un diálogo entre los avances tecnológicos y la demanda que comenzó a ofrecer el mercado.

Harían falta muchos trabajos como este para abarcar una simple aproximación de la historia de las ciencias de la computación. Parte de este contenido está basado en lo que aparece en el libro "Breve historia de la computación" (Coello Coello, Carlos A., 2004). A continuación se enumerarán, de modo superficial, los hechos más relevantes para nuestro objetivo. No servirán para aumentar el entendimiento pero sí para cimentar el hecho de que estamos hablando, pese a su relativa juventud, de un sector muy profundo en historia y complejidad:

- 500 a.C.: Aparición del ábaco como primera herramienta de cálculo. No es relevante por su parecido a un ordenador, pero se trataba de un artilugio que, utilizando un determinado método, facilitaba ciertas tareas. Podría decirse que utilizaba un algoritmo determinado para realizar cálculos.
- 1937: Se desarrolla el *Atanasoff Berry Computer*. Es el primer ordenador electrónico.
- 1945: Un insecto se introduce en una de las válvulas el ordenador Mark I y lo paraliza. A partir de ese momento la palabra *bug* (término inglés para decir "insecto") se utilizará describir cualquier problema que impida el funcionamiento de un programa informático.
- 1953: Se extiende el uso del lenguaje ensamblador.
- 1964: Se crea el lenguaje de desarrollo de alto nivel *BASIC* (siglas de *Beginner's All-purpose Symbolic Instruction Code*).
- 1970: Se crea el lenguaje de programación *PASCAL*. Lenguaje creado para facilitar el aprendizaje.
- 1976: Steve Jobs junto a Steve Wozniak fabrican su primer ordenador y crean *Apple*.
- Septiembre de 1982: Se comercializa el *Commodore 64*. Vende entre 17 y 22 millones de unidades.

- 1990: Se crea el lenguaje de documentos *HTML*. Hoy es ampliamente conocido por se el lenguaje en que se elaboran las páginas Web.
- 1991: Se populariza la programación orientada a objetos.

Como puede observarse, la historia de la computación abarca gran parte de la historia reciente. La lista anterior es un extracto de determinados hechos que, aun no siendo trascendentes para el lector y su comprensión, deben hacer ver su amplitud temporal para aumentar las expectativas y el calado que el lector reciba de este trabajo.

5.2.2. Antecedentes en la enseñanza

En este apartado trataremos de resumir las principales apariciones que la programación ha podido tener a lo largo de la enseñanza tradicional.

La presencia de la programación en los programas educativos se reduce, principalmente, a las carreras universitarias. Es en este ámbito donde, desde hace relativamente poco (en comparación con otras áreas clásicas), se ha venido desarrollando una actividad más intensa.

Es cierto que, en España, las nuevas tecnologías forman una parte muy importante en el contenido curricular en la etapa pre-universitaria. Pero también es cierto que, dado la ambigüedad en su descripción, la aplicación de una u otra forma termina recallendo en el interés particular del profesor que pretenda impartir una asignatura relacionada con la informática.

En mi experiencia como alumno de instituto pude cursar la asignatura de informática a la edad de 16 años (alrededor del año 2000). El contenido de la misma fue bastante (quizá demasiado) amplio y el profesor trató de abarcar contenido tan dispar como: procesamiento de textos; hojas de cálculo; bases de datos; programación en *C*; programación en *Logo*; mecanografía…

Lejos de criticar aquella experiencia sólo puedo agradecer el esfuerzo de aquel profesor. Dejando a un lado lo disparatado que puede resultar tratar de impartir ese conocimiento en un grupo tan heterogéneo de alumnos, para muchos supuso el primer contacto con determinados conceptos, procesos mentales y razonamientos. Aquel profesor transmitía interés por lo que enseñaba y, visto desde el punto de vista que nos trae en este trabajo, dejaba en evidencia lo que decía más arriba: la enseñanza de conceptos tan concretos como la programación o razonamiento lógico dependían, en la enseñanza tradicional de nuestro país, del gusto particular del profesor encargado de impartirla.

La programación, o el pensamiento computacional en su parte más amplia, queda relegada a una enseñanza superior (profesional o universitaria) y no aparece en la educación a edades tempranas. Podemos decir, concluyendo, que el panorama histórico ha sido bastante hueco por lo que habrá que esforzarse en recorrer el largo camino que se nos presenta.

5.3. Justificación, conocimientos y capacidades adquiridas

Este documento centra su objetivo en la programación y sus posibles efectos en las habilidades que mejora su implantación en la enseñanza de los niños.

Lo que esperamos de un niño cuando aprende a programar no es, únicamente, que conozca la sintaxis de un determinado lenguaje de programación. Lo que esperamos de él es que sea capaz de identificar un problema, analizar la situación, proponer una solución y, ahora sí, materializarla utilizando herramientas asequibles y a su alcance.

El proceso previo a esa materialización es lo que buscamos fomentar. Se trata de desarrollar un concepto mucho más amplio que la programación, donde buscamos mejorar una serie de habilidades analíticas, creativas y resolutivas. Buscamos desarrollar, tal y como describiremos a continuación, el "pensamiento computacional".

5.3.1. Pensamiento computacional

El pensamiento computacional es "la integración de la potencia del pensamiento humano con las capacidades de los ordenadores." (Phillips, Pat, 2009)

En el año 2006 la profesora habló sobre el pensamiento computacional en un seminario. Lo definió como una forma de "resolver problemas, diseñar sistemas, y entender el comportamiento humano haciendo uso de los conceptos fundamentales de la informática." (Wing, J.M., 2006)

En este seminario, Wing comentó que el pensamiento computacional conlleva los siguientes conceptos:

- Descomposición de problemas.
- Representación y modelado de información.
- Búsquedas binarias.
- Recursividad.
- Paralelización.

Según la International *Society for Technology in Education* (*ISTE*) y la Computer Science Teachers Association (*CSTA*), citados en el artículo "Computational Thinking: A Digital Age Skill for Everyone" (Barr, David, Harrison, John y Conery, Leslie, 2011), el pensamiento computacional es un proceso de resolución de problemas que incluye, entre otras, las siguientes características:

- La formulación de problemas de forma que nos permita la utilización de ordenadores y otras herramientas para su resolución.
- Análisis y organización lógica de la información.
- Representación de la información mediante modelos y simulaciones.
- Automatización de soluciones mediante el pensamiento algorítmico.
- Identificación, análisis e implementación de soluciones posibles con el objetivo de conseguir la combinación más eficiente y efectiva de pasos y recursos.
- Generalización y transferencia de este proceso de resolución de problemas a una amplia variedad de situaciones.

En este mismo artículo se definen una serie de habilidades que sustentan estas características y que son esenciales en torno al pensamiento computacional:

- Confianza en el tratamiento de la complejidad.
- Perseverancia ante el trabajo en problemas difíciles.
- Tolerancia a la presencia de ambigüedad.
- La habilidad para tratar con problemas sin una solución única.
- La habilidad de comunicarse y trabajar con otros para conseguir un objetivo común o solución.

Como veremos a continuación, hay conceptos cercanos a lo que describiremos como aptitudes desarrollables ante el aprendizaje de un lenguaje de programación.

Programar no es pensar computacionalmente, pero sí sirve para desarrollar estas destrezas que serán aplicables a otros sectores.

5.3.2. Habilidades de aprendizaje y programación

Basaremos las habilidades relacionadas e impulsadas mediante la programación en el informe “El aprendizaje para el siglo 21” (Rusk, Natalie, Resnick, Mitchel y Maloney, John, s.f.)

Según este informe, existen nueve tipo de habilidades de aprendizaje que son claves para conseguir cumplir las expectativas del futuro que nos espera. A continuación, se describirán esas nueve habilidades, agrupadas en tres grupos, y se relacionarán con aquello que nos facilita la enseñanza de un lenguaje de programación.

5.3.2.1. Habilidades relacionadas con la información y la comunicación

5.3.2.1.1. La información y la alfabetización mediática

Se define como "analizar, acceder, gestionar, integrar, evaluar y crear información en una variedad de formas y medios(...)".

En nuestro caso, la programación desencadenará una actitud analítica que no sólo se reflejará en la resolución del problema que se plantee en cada caso, si no que se asimilará y fomentará el contacto directo con todos los medios para la búsqueda de información y soluciones en base a los recursos disponibles.

5.3.2.1.2. Habilidades de comunicación

Se define como "entender, gestionar y crear una comunicación oral, escrita y multimedia en una variedad de formas y contextos."

La programación es, en sí misma, un nuevo método de comunicar ideas complejas. La asimilación de este nuevo modo de expresarse desarrollará, en la medida de lo posible, las habilidades comunicativas del individuo.

5.3.2.2. Habilidades para pensar y resolver problemas

5.3.2.2.1. Pensamiento crítico y sistémico

Se trata de fomentar la toma compleja de decisiones y la comprensión de la interconexión entre distintos sistemas.

La programación y resolución algorítmica entrena estos puntos en la medida que el problema sea complejo en sí. El pensamiento crítico será una parte fundamental para poder obtener soluciones eficaces.

5.3.2.2.2. Identificación de problemas, formulación y solución

A través del proceso de programación (y no refiriéndonos sólo al proceso de escritura en un determinado lenguaje) se fomentan cada uno de estos puntos.

La programación culmina con un algoritmo y su implementación (la solución) pero antes se ha tenido que identificar el problema, analizarlo y abstraerlo.

5.3.2.2.3. Curiosidad intelectual y creativa

Se define como "el desarrollo, la implementación y la comunicación nuevas ideas a otros, en un estado abierto y sensible a nuevas perspectivas."

La programación en sí misma, y una vez asimilada más todavía, supone un desafío constante para el programador. No se trata únicamente de un ámbito con un dinamismo atroz y tecnológicamente cambiante, si no que, una vez interiorizado su poder, cualquier problema supone un reto apetecible. La satisfacción ante una solución suele proporcionar al programador un gran bienestar.

5.3.2.3. Habilidades Interpersonales y Autonomía

5.3.2.3.1. Habilidades interpersonales y colaborativas

En este habla sobre la capacidad para trabajar en equipo y con liderazgo. Se refiera a habilidades tales como la empatía, a adaptabilidad y la capacidad para respetar otras opiniones.

En lo que aquí nos atañe, la programación te va guiando sobre estas pautas, si bien no en todos los casos, muchas veces a la fuerza. La programación se puede convertir en una tarea solitaria y esto, casi siempre, terminará en un bloqueo mental y, finalmente, en la reafirmación de que cualquier problema se soluciona mejor si se lo cuentas a alguien. El ego termina desapareciendo a favor de los buenos resultados.

5.3.2.3.2. Autonomía

Se trata de conocer las propias necesidades, nuestros puntos fuertes y débiles.

La programación provoca un aprendizaje constante. Hoy en día los recursos están a disposición de todos y, dada la complejidad y dinamismo en la que se ubica esta tarea, la autonomía será fundamental si se quiere conseguir algún progreso. La programación exige grandes dosis de autonomía para no quedarse obsoleto y conseguir nuevas motivaciones.

5.3.2.3.3. Responsabilidad y adaptabilidad

Se deberá respetar la diversidad y se deberá ser capaz de adaptarse a cualquier contexto.

Programar exige tareas en equipos multidisciplinares. Esto provoca que haya que dirigirse a públicos muy diversos en las distintas etapas de desarrollo de programas.

5.3.2.3.4. Responsabilidad social

La programación puede suponer un contexto divertido para el planteamiento de problemas de esta índole.

5.3.3. El papel de los profesores

Los profesores deberán fomentar que los estudiantes "piensen computacionalmente haciendo que los proyectos tecnológicos pasen de ***usar*** las herramientas y la información a ***crear*** las herramientas y la información." (Phillips, Pat, 2009)

El gran problema es que, para que esto suceda, los profesores deberán pasar (sufrir en muchos casos) un proceso de actualización personal que los habilite en estas tareas.

Como dice Mark Clarkson, director de la IT and Computing at Egglescliffe School in Teeside:

> Había una época donde los departamentos de tecnología de las escuelas estaban formados por profesores de educación física que recopilaban los resultados de sus equipos de fútbol en hojas de cálculo y un par de profesores de matemáticas o física que sabían algo sobre programación. (…) Tiene que haber una formación masiva de profesores para asegurar que son capaces de proporcionar un contenido distinto al que estaban acostumbrados a proporcionar. (…) Hemos creado una generación de profesores con un conjunto de habilidades que no es lo que necesitaremos a largo plazo para el futuro de la informática en las escuelas. (Mark Clarkson, citado en O'Kelly, Lisa, 2012)

5.4. Aproximaciones reales

5.4.1. Situación actual

Llegados a este punto toca mirar a nuestro alrededor y detectar movimientos y tendencias.

España no vive una situación muy apta para inversiones en materia de innovación pedagógica. Cierto es que estos avances no requieren una partida económica de gran alcance pero, cierto es también, que el foco social no está preparado, en medio de esta crisis, para encajar este tipo de propuestas.

Pese a todo he podido encontrar casos concretos de personas e instituciones que, movidas en mayor parte por la pasión y la confianza, llevan a cabo determinadas acciones destinadas a inculcar un sentimiento de utilidad en torno a esta idea. Estos casos los trataremos en el siguiente apartado.

Por otro lado, en países cercanos sí que hay una corriente que está teniendo bastante calado en las instituciones políticas que, en última instancia, son los que tienen que entender estas propuestas y su posible calado en el futuro.

En este sentido nos estamos refiriendo, principalmente, al Reino Unido. En el siguiente apartado trataremos ejemplos y manifestaciones de alcance que están surgiendo en el país anglosajón.

5.4.2. Ejemplos

Comenzaremos con el caso de más peso y calado. Me refiero al movimiento británico que cada vez está adquiriendo más importancia. Se trata de una corriente de personalidades y medios que reclaman medidas a los estamentos políticos para que tomen cartas en el asunto y modifiquen algunos aspectos de la enseñanza para que los niños aprendan a programar en las escuelas.

En este sentido hay que comentar la particular cruzada que el diario *The Guardian* está llevando a cabo mediante la publicación de una serie de artículos en su medio digital (http://www.guardian.co.uk/education/computerscienceandit). En estos el medio británico trata, en torno a la programación, temas como los siguientes:

- La necesidad de una revolución en las aulas.
- La visión de los profesores.
- Ideas para el cambio.
- Opiniones de expertos.

En esta serie de artículos se comentan cosas tan interesantes como las siguientes:

> Lo que olvidamos fue que los coches no mueven el mundo, monitorizan nuestras comunicaciones, alimentan nuestros móviles, gestionan nuestras cuentas bancarias, guardan nuestros diarios, administran nuestras relaciones sociales, cotillean nuestras actividades personales e, incluso, cuentan nuestros votos. Son las computadoras las que hacen todo esto y mucho más. (Naughton, John, 2012)

También comentan un punto de partida en cuanto a determinados conceptos que deben aprender los niños en esta aproximación a la programación: algoritmos, criptografía, inteligencia artificial, biología computacional, búsquedas, recursión, y heurística.

Hay que mencionar también la figura de John Naughton. Se trata de un académico británico que ha redactado un "manifesto" (http://www.guardian.co.uk/education/2012/mar/31/manifesto-teaching-ict-education-minister) dirigido a Michael Gove (secretario de estado para la educación) para solicitar la inclusión de las ciencias de la computación como parte del currículo escolar.

Estas y otros movimientos de gran calado social son los que están poniendo a los británicos a la cabeza de la carrera tecnológica que, desde un análisis superficial y no demasiado complejo, lo pondrá a la cabeza en otros muchos más aspectos de aquí a unos años.

Por otro lado, en España no se está percibiendo una corriente parecida. Sí que hemos disfrutado de una serie de campañas a nivel estatal en las que se fomentaba el uso de las nuevas tecnologías aunque, más que como fin en sí mismo, como facilitador de las labores pedagógicas.

Después de una labor de rastreo he conseguido identificar determinadas figuras y movimientos particulares que sí están interesados en estos temas pero que, en la mayoría de los casos, o bien no han contado con el necesitado apoyo por parte de las autoridades, o bien no han recibido la difusión correcta que ampliara su alcance a una mayor parte de la sociedad.

En cualquier caso, citaremos también un par de ejemplos desde dentro de nuestras fronteras. No están tan respaldados y promocionados como los que podemos encontrar en otros países con iniciativas similares, pero sí sirven para detectar que hay una pequeña comunidad de personas dispuestas a innovar en este sentido.

Primero citaré el proyecto "Diviértete programando" (http://www.aletic.es/archivos/tic/1246352824.pdf) que se llevó a cabo durante el curso 2008-2009 en la provincia de León. Se trata de una iniciativa nacida de la Asociación de Leonesa de Empresas TIC. Durante un período determinado, niños entre 9 y 12 años de más de cincuenta centros de León tuvieron la oportunidad de aprender a programar utilizando el lenguaje KPL (Kid's Programming Languaje). Este proyecto fue capaz de mejorar capacidades tales como la abstracción o la inferencia lógica. Además, según la percepción de los padres, se obtuvieron resultados muy positivos y estos se mostraron muy propensos a repetir experiencia.

Otro ejemplo cercano sería el denominado *Scratch day 2012* (http://day.scratch.mit.edu/event/578). Se trata de un evento a nivel internacional secundado en ciudades de todo el mundo. En él se trata de dar a conocer Scratch como entorno de programación libre enfocado a personas que no tienen conocimientos de programación.

Estos son sólo unos cuantos ejemplos cercanos que podemos encontrar. El panorama no es muy positivo en nuestro país pero, tal y como hemos visto, hay iniciativas en otros lugares del mundo que seguro terminarán influyéndonos de una u otra forma.

5.4.3. Futuro y tendencias

Tal y como cerraba el punto anterior, el panorama español no es excesivamente positivo en cuanto al futuro de la programación en las aulas. Nuestra tarea radicarán en inculcar la necesidad en parcelas muy concretas e intentar que "cale" en la comunidad educativa.

También, según lo ya visto, podemos inferir que el panorama en Reino Unido, si bien todavía no está terminado, tienen un punto de partida muy propenso a mejorar. La comunidad educativa y personalidades relacionadas se encuentran dentro del camino y trabajan conjuntamente en su reclamación de estos contenidos para los más pequeños.

Tal y como podemos leer en "Young coders: what's happening in the rest of the world?" (2012), hay otros países con iniciativas en este camino. Tal es el caso de

Escocia que se encuentra en plena reforma del contenido educativo en el que ya se está incluyendo la computación en edades hasta los 14 años.

Estados Unidos también se encuentra en un proceso de mejora. Tal y como hemos visto hay grandes movimientos en torno a la idea de inclusión de conceptos educativos como el pensamiento computacional.

En la India la educación en computación es una asignatura desde los 14 años en adelante. Y países como Korea del Sur ya tenían una larga tradición en la enseñanza de computación en las escuelas.

5.5. Herramientas y recursos didácticos

5.5.1. Introducción

En este apartado analizaremos varias herramientas útiles para enseñar a programar. No son todas y, tal vez, tampoco las mejores. Sí opino, de cada una de ellas, que son las más especiales en el momento que se escribe este texto. A continuación las analizaremos de forma que pueda entenderse su aportación y características particulares.

5.5.2. Logo: un clásico

Se trata de un lenguaje de programación de alto nivel creado en 1967 por Danny Bobrow, Wally Feurzeig y Seymour Papert.

Es un lenguaje muy fácil de aprender, lo que hace que sea idóneo para su utilización con niños. Además, es lo suficientemente potente para enseñar los principales conceptos de la programación dado que soporta manejo de ficheros, listas y soporte para entrada/salida.

Una característica que se utiliza bastante en la enseñanza es producir "gráficos tortuga". Esto quiere decir que iremos proporcionando instrucciones para mover el cursor (la tortuga) para que se vaya moviendo y, a su vez, dibujando la figura deseada.

A continuación, un ejemplo:

```
para cuadrado
repite 4 [avanza 50 giraderecha 90]
fin
cuadrado
```

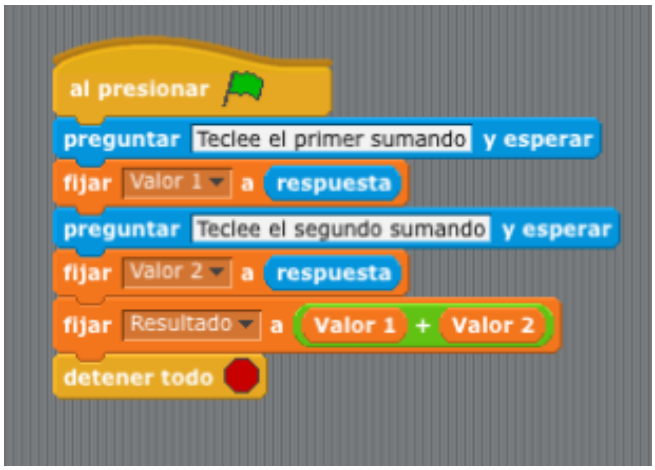
En este trozo de programa lo que estamos es diciéndole al ordenador que cuando le escriba la palabra "cuadrado" repita cuatro veces la acción de avanzar 50 puntos y girar a la derecha 90º. Si visualizamos mentalmente estas acciones (o las simulamos con un lápiz y trozo de papel) comprobaremos que estamos en lo cierto.

5.5.3. Scratch: descripción y ejemplos

Scratch es una aplicación informática creada en 2007. Utiliza el lenguaje de programación Logo para que los usuarios puedan experimentar y explorar determinados conceptos relacionados con la programación. Fue creado por el Es desarrollado por el *the Lifelong Kindergarten group* en el *Media Lab* del *MIT* (*Massachussets Institute of Tecnology*).

Una característica muy acorde con las últimas tendencias es que, una vez creada una cuenta en su página Web, los usuarios que creen un "programa" con Scratch, podrán compartirlo de forma que cualquier persona pueda descargárselo y probarlo en su propio ordenador.

A continuación, de forma muy básica, se ilustra un ejemplo de programa creado con Scratch:



Como se puede observar, es un entorno totalmente gráfico y muy llamativo. Su estética es muy acorde a cualquier aplicación diseñada para los niños.

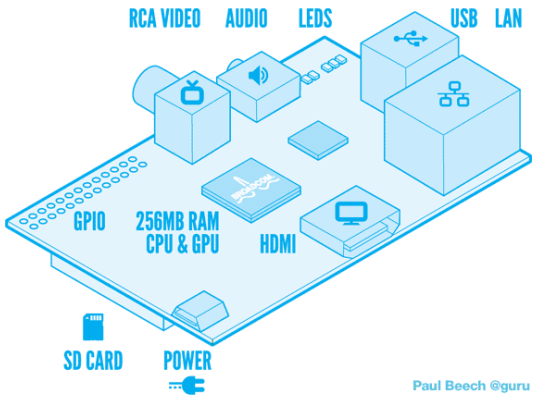
En el ejemplo expuesto se pretende hacer un programa que haga una suma de dos números. Para ello, y siguiendo el orden de las instrucciones, el programa pedirá un valor, luego otro y, finalmente, después de realizar la operación, mostrará el resultado.

No requiere mucha explicación el ejemplo en cuestión, aunque no está de más decir que esto es sólo una muestra muy básica frente a el potencial visual que se puede conseguir con esta aplicación.

5.5.4. Raspberry Pi

Este producto es el que más impresión me ha causado por lo ingeniosa que me parece su idea.

Se trata de una placa de ordenador de bajo coste desarrollada en el Reino Unido. Contiene todo lo básico para la ejecución de un sistema operativo y poder hacer pequeños desarrollos en él. En la siguiente imagen esquemática podemos ver si estructura:



Como se puede observar contiene una salida de audio, una salida de video, puertos usb, puerto de red, ranura para tarjeta de memoria SD y 256MB de memoria RAM. Es más que suficiente para que un grupo de niños aprendan, no sólo fundamentos de programación, si no fundamentos de arquitectura de computadores que de otro modo serían tan complejos y caros que resultaría muy difícil su acercamiento a las aulas.

No se trata de una iniciativa con ánimo de lucro. En realidad, tras esta gran idea hay una asociación caritativa cuyo objetivo es, según ellos mismos dicen, “promover el estudio de las ciencias de la computación y temas relacionados, sobre todo a nivel escolar, y para recuperar la diversión de aprender computación”.

En la imagen de la derecha podemos observar un ejemplo de esta placa funcionando y ejecutando un sistema operativo Linux plenamente funcional:

5.5.5. Computer Science Unplugged

Muy económico y, no por ello, menos útil y original es el siguiente recurso que voy a tratar. Se trata de un portal Web disponible de forma gratuita en el que podremos encontrar ejercicios que servirán para enseñar ciencia computacional de forma que no necesitemos el ordenador.

Se basa en la utilización de pinturas, puzles, cuerdas y cualquier cosa que esté al alcance para la asimilación de conceptos como los números binarios o los algoritmos.

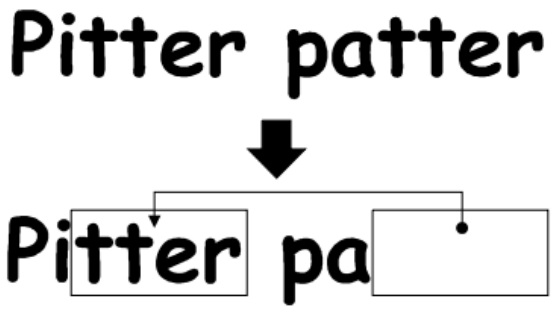
En su portal ponen a disposición de los usuarios un libro gratuito en el que tratan actividades en torno a la idea descrita. Este libro está organizado por conceptos a enseñar. Para cada uno pone un ejercicio que desarrolla esa idea. Siguiendo este patrón mostraremos cómo trata de enseñar el concepto de "compresión" mediante un ejercicio práctico:

Introduction

Computers have to store and transmit a lot of data. So that they don't have to use up too much storage space, or take too long to send information through a modem connection, they compress the text a bit like this.

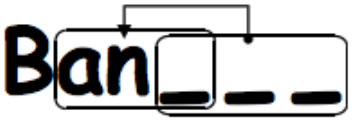
Demonstration and Discussion

Show "The Rain" OHP (page 25). Look for the patterns of letters in this poem. Can you find groups of 2 or more letters that are repeated, or even whole words or phrases? (Replace these with boxes as shown in the diagram below.)



Este contenido está extraído directamente del libro. En él podemos leer (en inglés) cómo introduce el concepto de compresión para, posteriormente, plantear ejercicios en torno a esta idea:

How would you solve this puzzle?



6. Conclusión

Queda demostrado que el tema central de este trabajo queda respaldado por una serie de hechos y movimientos que se han identificado como resultado de la investigación.

Hemos hecho una rápida pasada por los principales conceptos que son necesarios asimilar para adentrarse en el mundo de la programación enfocado al mundo de la enseñanza.

Tal y como hemos visto hay estudios respaldados por una comunidad de docentes cada vez más amplia. También hemos podido comprobar que estas ideas están creando cierta conciencia social en lugares como el Reino Unido.

Hemos tenido oportunidad, también, de conocer algunas de las principales herramientas que, de forma gratuita, podemos encontrar a nuestra disposición para sembrar la curiosidad de la programación y configurar un contenido curricular apto para los niños y que fomente las habilidades aquí nombradas.

Cuando elegí este tema para el trabajo lo hice por afinidad con mis gustos por la informática. Trataba de demostrarme a mí mismo que hay mucho camino que recorrer en torno a la educación en materia de programación. Uno de los descubrimientos que me ha gustado más de este trabajo ha sido el libro *Computing Science Unplugged* (Bell, Tim, Witten, Ian H. y Fellows, Mike, 2006). Me parece que capta la esencia de parte de lo que este trabajo quiere transmitir. No únicamente hay que programar. Debemos enseñar a pensar como lo hace un buen programador y, de ahí, extraer patrones de razonamiento que el alumno pueda aprovechar en un futuro y otras áreas. Debemos enseñar, no necesariamente con ordenadores, a tener un pensamiento computacional.

En resumen, hemos hecho un barrido a muy alto nivel que nos ha servido para confirmar, tal y como se esperaba al comienzo de la realización de este trabajo, que la programación (o pensamiento computacional en su vertiente más amplia) será completamente necesaria para que nuestra sociedad esté a la altura de las nuevas necesidades que vendrán en el futuro. Serán la llave para preparar a nuestros niños para profesiones que todavía no conocemos. Asentará las bases de un nuevo modo de aprendizaje y creará un caldo de cultivo que se centrará en qué crear, más a allá de plantearse, cómo crearlo.

Será indispensable para competir con las sociedades del futuro.

7. Referencias bibliográficas

7.1. Bibliografía

Barr, David, Harrison, John y Conery, Leslie (2011). “Computational thinking: A Digital Age Skill for Everyone”. Learning & Leading with Technology magazine March/April 2011 pp. 20-24.

Cerrada Somolinos, José Antonio y Collado Machuca, Manuel E. (2010). “Fundamentos de programación”. Editorial Universitaria Ramón Areces; Edición: 1 (14 de julio de 2010). España.

Coello Coello, Carlos A. (2004). “Breve historia de la computación”. Fondo de Cultura Economica USA. España.

Wing, J.M. (2006). “Computational Thinking,” Communications of the ACM, vol. 49, no.3, pp. 33-35

7.2. Fuentes electrónicas

Bell, Tim, Witten, Ian H. y Fellows, Mike (2006). "Computer Science Unplugged". Recuperado el 27 de abril de 2012, de http://csunplugged.org/sites/default/files/activity_pdfs_full/CS_Unplugged-en-10.2006.pdf

Naughton, John (2012). "Why all our kids should be taught how to code". Artículo periodístico. Recuperado el 16 de abril de 2012, de http://www.guardian.co.uk/education/2012/mar/31/why-kids-should-be-taught-code

O’Kelly, Lisa (2012). “Young coders: the teachers' view”. Artículo periodístico. Recuperado el 17 de abril de 2012, de http://www.guardian.co.uk/education/2012/mar/31/computer-science-teachers-training-school

Phillips, Pat (2009). “Computational thinking. A problema-solving tool for every classroom”. Recuperado el 19 de abril de 2012, de http://education.sdsc.edu/resources/CompThinking.pdf

Rusk, Natalie, Resnick, Mitchel y Maloney, John (s.f.). "21st Century Learning Skills". Mit Media Lab. Recuperado el 20 de abril de 2012, de http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/docs/Scratch-21stCenturySkills.pdf

"Young coders: what's happening in the rest of the world?" (2012). Artículo periodístico. Recuperado el 16 de abril de 2012, de http://www.guardian.co.uk/education/2012/mar/31/young-coders-rest-of-world

7.3. Bibliografía recomendada

"Center for Computional Thinking Carnegie Mellon" (s.f.). Sitio Web oficial. Recuperado el 23 de abril de 2012, de http://www.cs.cmu.edu/~CompThink/

"Enseñar a programar a los niños. Una brevísima introducción al acto de programar (o a los mecanismos de pensamiento necesarios para programar)" (2009). Debate en el portal educativo del estado argentino. Recuperado el 18 de abril de 2012, de http://portal.educ.ar/debates/educacionytic/inclusion-digital/ensenar-a-programar-a-los-nino.php

"Guía de inicio Scratch" (2007). Por el Lifelong Kindergarten Group. Recuperado el 22 de abril de 2012, de http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/files/file/translated-docs/GuiaDeInicioScratch.pdf

Heng, Christopher (2011). "Free Educational Programming Languages / Free Children's Programming Languages". Recopilación de herramientas gratuítas. Web de recursos para programadores. Recuperado el 22 de abril de 2012, de

http://www.thefreecountry.com/compilers/educational-programming-languages.shtml

Holowczak , Richard (2007). "Programming Concepts. A Brief Tutorial for new Programmers". Zicklin School of Business. Recuperado el 19 de abril de 2012, de http://cisnet.baruch.cuny.edu/holowczak/classes/programming/

"Raspberry Pi FAQs" (2012). Sitio Web oficial. Recuperado el 19 de abril de 2012, de http://www.raspberrypi.org/faqs

Rushkoff, Douglas (2010). “Program or Be Programmed” (Edición Kindle). OR Books.