16

NOMBRE DE CLASE:

Programación bajo presión

Duración: 45–60 minutos : Preparación: 5 minutos

Meta: Hacer hincapié en la importancia de verificar el trabajo realizado y la escritura de programas en la secuencia correcta.

RESUMEN:

Esta lección tomará una idea de una actividad anterior (4. Programación en hoja cuadriculada) y el uso de esas habilidades para hacer hincapié en la importancia de completar los programas que funcionan con una determinada secuencia y la importancia de comprobar frecuentemente la existencia de errores (bugs) en los programas.

OBJETIVO:

Los estudiantes —

- Aprenderán a comprobar su trabajo y el de otros
- Pensarán sobre secuencias
- Practicarán imaginarse resultados esperados
- Practicarán realizar "tareas de pensamiento" bajo presión

Materiales:

- Kit de dibujos y algoritmos de ejemplo de la clase 4
- Tarjetas de programación
- Una hoja grande de papel cuadriculado
- Fichas u hojas de papel en blanco
- Marcadores, lapiceras o lápices (dos o tres colores)

PREPARACIÓN:

Esto funciona mejor si se aprendieron los detalles de la actividad de programación en papel cuadriculado.

Imprimir el kit de dibujos y algoritmos para cada grupo.

Imprimir las tarjetas de programación para cada grupo.

Imprimir varias cuadrículas de dibujo dependiendo de la versión de ejercicio que estés haciendo.

VOCABULARIO:

Errores ("bugs" en inglés)—Problemas en el código de los programas

Depurar ("debugging" en inglés)—Arreglar problemas en el código

Secuencia—El orden en que son hechas las cosas

REPASO:
La intención de este repaso es recordar lo visto en la clase anterior. Si cubres las actividades en distinto orden, por favor realiza tu propio repaso aquí.

Preguntas para la participación en clase:
- ¿Qué hicimos en la clase anterior?
- ¿Recuerdas lo que es un parámetro?
- ¿Es un parámetro también una variable? ¿Por qué sí y por qué no?

Esta lección hace hincapié en la importancia de completar los programas que funcionan con una determinada secuencia y la importancia de comprobar frecuentemente la existencia de errores (bugs) en los programas.

DESARROLLO:

La actividad es clave aquí. La mayoría de las habilidades necesarias para jugar este juego fueron explicadas en la clase 4. La experiencia en esta actividad consiste en poner a prueba las habilidades aprendidas.

Los informáticos se enfrentan siempre a plazos para terminar su trabajo. A medida que el tiempo se acaba, el programador es tentando a saltear pasos importantes para la comprobación de la calidad de su trabajo, o relegar estas pruebas cuando todo el trabajo ha sido terminado. Para simular la presión de trabajo en estas situaciones, esta clase está cronometrada.

Dividir la clase en grupos de 4 a 6 y alinearlos en tandas hacia un costado del aula (jugar afuera lo hace más excitante). Del otro lado del aula (o el patio), colocar un papel cuadriculado por cada tanda de chicos. Deja una hoja en blanco cerca de cada imagen.

Las reglas son simples. Cada equipo debe mandar el primer estudiante de la fila a tomar una hoja de papel cuadriculado y dibujar un primer símbolo (letra) de programación en la hoja en blanco al lado de la imagen. El alumno regresa hacia la fila y toca en la mano al siguiente compañero. El siguiente alumno va hacia los papeles, mira la imagen, revisa la programación de los estudiantes anteriores, y añade otro símbolo. Si un alumno encuentra un error en el programa del grupo, debe arreglar el código escrito en lugar de añadir otro símbolo. Este proceso se repite hasta que el grupo sienta que ya terminó de programar la imagen correctamente. La velocidad y energía de este juego depende de si es jugada afuera o dentro del aula. Si el espacio es limitado, requiere que cada estudiante camine, o incluso pase el papel al escritorio de su compañero. La versión del juego en la que corren puede parecer más impresionante, dado que los alumnos tienen menos tiempo para procesar y comunicarse durante las transiciones.

Un ganador es declarado cuando todo el equipo cree que ya terminó, y el docente chequea la validez del algoritmo para recrear el dibujo original. Este juego puede ser jugado de nuevo con múltiples dibujos y múltiples variaciones.

Cuando el juego termine, reunir a los alumnos y preguntarles qué aprendieron.

- **¿Fue fácil crear el código perfecto cuando deben trabajar tan rápido?**
- **¿Qué tan fácil o difícil fue leer el código del grupo que ya fue escrito?**
- **¿Encontraron algún error? ¿Cómo sabían que eran errores?**
- **¿Es más fácil o más complicado tener mucha gente involucrada en la creación del programa en tiempos diferentes?**
- **¿Se les ocurre algún truco para hacer el trabajo más fácil para la persona que les sigue?**
- **¿Qué deseas que haga la persona que vino antes que haga que se agilice tu trabajo? ¿O qué hizo la persona para que tu trabajo sea más rápido y preciso?**

AJUSTES:

EDAD:

Hasta los 7 años: Puedes tener estudiantes muy jóvenes que recreen una figura simple hecha de bloques. En un clima apresurado, esto puede ser más manejable. Se trata menos sobre el "algoritmo" pero todavía mantiene las ideas que queremos presentar.

De 8 a 10 años: Esta actividad puede ser más o menos escrita.

De 11 a 12 años: Añade más complejidad con dibujos complejos o con colores adicionales. Decide si quieres que usen las "funciones" de la clase 4.

VELOCIDAD:

Rápido: La carrera es caótica e hilarante. Dale una oportunidad si cuentas con grandes espacios abiertos.

Intermedia: Es ideal para un aula. Tienes a los estudiantes caminando a lo largo del aula. Pueden tener requerimientos adicionales que distraen a los alumnos mientras intentan completar su tarea, como caminar de espaldas, requerir que dejen una mano en el escritorio todo el tiempo, o tener que leer todo el programa antes de empezar a escribir.

Lento: Los estudiantes se mantienen sentados (en filas, círculos, o grupos pequeños) y pasan la imagen y la hoja de banco en banco. No permitas que los estudiantes se pasen pistas... sólo el que posee las hojas tiene permitido hablar.

DIFICULTAD:

Difícil: Tienes dos dibujos en cada momento, y los estudiantes deben determinar cuál está siendo programado mientras trabajan en el algoritmo.

Fácil: La persona que programa se queda al lado de las hojas y ayuda a la siguiente persona. Esto añade una continuidad mientras impide que los estudiantes se pierdan fácilmente.