Propuesta de planificación anual para Tecnologías de la Información, 3º año de la NES (TI3), CABA

Versión del 15 de febrero de 2017[1]

Índice

Historial de versiones

Versión	Cambios
15/02/2017	Versión inicial con la planificación completa y secuencias didácticas de las clases 1 a 8.
24/02/2017	Se agregarán las secuencias didácticas de las clases 9 a 19.
03/03/2017	Se agregarán las secuencias didácticas de las clases 21 a 30.

Prólogo

Es complejo entender el mundo en el que vivimos sin cierto conocimiento acerca de cómo funcionan las computadoras y los programas que las comandan. Interactuamos con ellas todo el tiempo: cuando buscamos en Internet, cuando chateamos mediante nuestros celulares y también cuando pagamos el boleto en el transporte público, o cuando mediante el dispositivo conocido como "pedal" activamos el sistema de frenado de un auto. La tecnología permea nuestra vidas y las discusiones que la rodean van ganando lugar en la agenda pública. Si no conocemos de estos temas, nuestra posibilidad de participar como ciudadanas y ciudadanos plenos se ve reducida. Por eso es tan importante que la escuela aborde las Ciencias de la Computación y en consecuencia la importancia de la materia Tecnología de la Información.

Ciencias de la Computación es el nombre que recibe la disciplina que estudia cómo funcionan las computadoras y los programas que las controlan, cómo se comunican éstas entre sí, y tantas otras cuestiones relacionadas con la programación, la construcción de sistemas y el procesamiento de información de todo tipo.

La materia **Tecnología de la Información** es una de las formas en que la escuela argentina empieza a responder a la necesidad de enseñar Ciencias de la Computación, y de ahí su importancia. La Iniciativa Program.AR viene trabajando desde hace varios años para contribuir con la llegada de las Ciencias de la Computación a la escuela, y por ende desea contribuir a que esta materia sea un éxito. Por ese motivo, ha puesto su equipo de profesionales a trabajar en una propuesta de trabajo para el aula. Esta primera edición está pensada para "TI3", es decir, la materia Tecnología de la Información de tercer año de la Nueva Escuela Secundaria de la Ciudad Autónoma de Buenos Aires.

Se trata de una **planificación de todo el año y sus correspondientes secuencias didácticas**, ajustadas al plan de estudio oficial de la materia, con el objetivo de brindarle al docente un marco sólido para tomar de base a la hora de pensar sus propias clases. En ellas se abordan temas de programación, organización y arquitectura de computadoras, sistemas operativos y seguridad informática.

Propone un **recorrido** original: comienza presentando nociones elementales de programación en una plataforma gráfica, apta para adolescentes sin ningún conocimiento previo, con el objetivo de explicar la noción de programa y una introducción a la de algoritmo. El paso siguiente es hacer algunos ejercicios muy sencillos en un lenguaje textual, para poder preguntarse qué contienen los programas ejecutables, aquellos a los que en nuestra experiencia cotidiana simplemente les hacemos doble click. Esa pregunta sirve de puntapié inicial para indagar sobre cómo funcionan las computadoras (su organización y su arquitectura), lo que nos permite transformar una noción cercana a una caja negra, casi mágica, en un concepto sobre el que podemos razonar.

Luego de esa "sumergida" en profundidad, volvemos a acercarnos a la "superficie" pasando por los conceptos de sistema operativo, de virus y de algunos otros relacionados con los aspectos de la seguridad informática que tienen contacto más directo con nuestro uso cotidiano de las

computadoras. Terminamos el año volviendo a la programación en entornos para principiantes, repasando y profundizando las nociones ya adquiridas y visitando algunas nuevas.

Este trabajo se pone a disposición de la comunidad bajo una **licencia *Creative Commons***[2] como una forma de incentivar la creación de obras derivadas. Dicho de otra forma, fomentamos activamente que las y los colegas generen sus propias versiones de este material y las compartan con la comunidad.

Consideramos que este material se encuentra en la categoría **trabajo en curso**, y que por lo tanto irá mejorando a medida que vaya siendo usado y discutido. Esperamos que las y los colegas lo encuentren útil y quedamos atentos a críticas o comentarios. Los esperamos en info@program.ar y los invitamos a revisar periódicamente nuestro sitio web, o seguirnos en las redes sociales, para mantenerse al tanto de las futuras versiones.

El equipo de Program.AR

---

www.program.ar

@Programar2020 | programar2020

[2] Específicamente, una licencia "Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional", cuyos detalles pueden consultarse en https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es.

Planificación TI3

Clase 1 - Introducción

Objetivos
Aplicar estrategias y técnicas para crear animaciones y videojuegos mediante entornos de programación educativa. Comprender el carácter específico de las instrucciones empleadas en programación. Realizar una primera aproximación a conceptos centrales para la realización de un programa.
Contenido y desarrollo
Introducción a la materia y al entorno Alice. Instalación de Alice. En esta clase se instalará y presentará el entorno Alice. Los componentes de Alice pueden presentarse como si fueran las partes necesarias para la realización de una película: la cámara, la iluminación, la locación, la escenografía, los actores y el guión. Ver Actividades.
Herramientas
Alice

Clase 2 - Noción de programa y proyecto simple I

Objetivos
Comprender la noción de programa y la diferencia entre tiempo de creación y de ejecución del mismo. Establecer relaciones entre el orden de las acciones programadas y los resultados obtenidos. Asumir posturas activas para la exploración de las herramientas, opciones y limitaciones que propone un entorno. Aplicar los conceptos adquiridos para desarrollar animaciones simples. Planificar y realizar un proyecto grupal de animación y/o juego que permita abordar alguna temática significativa para el grupo.

Contenido y desarrollo
Presentación de la noción de programa y secuencialidad. Apropiación del entorno y la herramienta. Se verá cómo utilizar los métodos disponibles en Alice, intercambiando el orden de las instrucciones y analizando los resultados al ejecutar el programa. Construcción de una animación simple. Presentación del proyecto. Ver Actividades.
Herramientas
Alice

Clase 3 - Métodos I

Objetivos
Introducir la idea de abstracción a través del uso de métodos en Alice. Reconocer cómo descomponer el problema/guión en acciones abstractas más abarcativas como estrategia para encarar la creación de un programa. Reconocer la función de los métodos.
Contenido y desarrollo
A partir del relato de una historia se pedirá a los alumnos que programen un guión. Debido a la longitud y repetición de acciones en la historia, la creación del programa se volverá tedioso y extenso. A posteriori, se introducirá como estrategia de resolución la identificación de acciones abarcativas que nuclean secuencias de instrucciones más simples dentro de la historia. Se identificarán esas acciones de manera conjunta y se procederá a reescribir el programa aprovechando el recurso de los métodos para plasmar las acciones más complejas y abstractas que componen la historia. Se identificarán las ventajas del uso de métodos al comparar los dos programas construidos, poniendo énfasis en la legibilidad, la extensión y claridad del programa resultante. Ver Actividades.
Herramientas

Alice; escenario inicial y guión de la actividad.

Clase 4 - Alternativa condicional

Objetivos
Comprender el concepto de alternativa condicional y de flujo de control alternativo o no secuencial. Utilizar correctamente los comandos necesarios para ejecutar alternativas condicionales. Realizar programas adecuados frente a escenarios cambiantes. Identificar el uso de alternativas condicionales tanto en situaciones cotidianas como en juegos y aplicaciones conocidas. Distinguir entre la programación de animaciones y la programación de juegos con interacción del usuario a partir del uso de eventos.
Contenido y desarrollo
Uso de condicionales. Interactividad. El objetivo es que los alumnos programen un juego en el que un personaje prenda todas las luces de una fila de casilleros, teniendo en cuenta que en ellos puede o no haber una luz. Este ejercicio servirá para introducir el concepto de alternativa condicional a partir de la instrucción if/else de Alice. Se buscarán ejemplos de toma de decisiones en la vida cotidiana, en juegos y aplicaciones para explicar el uso de alternativas condicionales. Ver Actividades.
Herramientas
Alice y el escenario inicial de la actividad.

Clase 5 - Proyecto simple II

Objetivos
Aplicar técnicas y estrategias para crear animaciones y videojuegos mediante entornos de programación educativos. Propiciar trabajo activo y colaborativo entre pares.
Contenido y desarrollo

Se trabajará sobre el proyecto presentado en la clase 2. Se presentará el nivel de avance de cada grupo y los obstáculos y objetivos a futuro. A partir de esto, se realizará un intercambio para abordar de manera conjunta los aspectos de la herramienta que requiera cada pieza. Ver Actividades.
Herramientas
Alice

Clase 6 - Variables I

Objetivos
Seleccionar y aplicar estrategias para la resolución de un problema simple. Comprender el concepto de variable y relacionarlo con la memoria y las formas de almacenamiento de las computadoras.
Contenido y desarrollo
Uso de variables e interactividad. Se mostrará como usar Alice para hacer programas interactivos. Como primer ejemplo se puede hacer un programa en el que el usuario ingrese su nombre, y algún personaje lo diga. Como ejercicio para introducir variables, puede solicitarse que ingresando una única vez el nombre del usuario todos los usuarios lo saluden. Se explicará luego el uso y el concepto de variable como un lugar donde se puede almacenar un dato y después recuperar esa información que está guardada. Ver Actividades.
Herramientas
Alice

Clase 7 - Diferencia entre programa y algoritmo

Objetivos
Reconocer la función de los algoritmos y su representación en pseudocódigo.
Contenido y desarrollo
Creación de programas sencillos en Alice para luego discutir el algoritmo que implementan y la diferencia entre programa y algoritmo. Los programas a implementar realizarán operaciones aritméticas sencillas y utilizarán variables y alternativas condicionales. Por ejemplo, determinar el máximo entre 3 valores. Ver Actividades.
Herramientas
Alice

Clase 8 - Lenguajes de programación

Objetivos
Reconocer diferencias y similitudes entre lenguajes y entornos de programación. Reforzar la idea de que un mismo algoritmo puede tener varias implementaciones, incluso en varios lenguajes.
Contenido y desarrollo
Implementación en C simplificado de un programa similar a los realizados en Alice la clase anterior. Comparación entre Alice y C para resaltar que hay similitudes independientemente del lenguaje elegido para la implementación. Ver Actividades.
Herramientas
Alice y compilador C (ya instalado en Huayra). Archivos disponibles en https://goo.gl/hqZRj0.

Clase 9 - El lenguaje de máquina

Objetivos
Diferencia entre lenguajes interpretados y lenguajes compilados. Reflexionar sobre la relación entre lenguajes de alto y bajo nivel. Analizar la función, estructura y funcionamiento de los lenguajes de bajo nivel.
Contenido y desarrollo
Partiendo de un programa sencillo en algún lenguaje textual (C), se presentará el resultado al desensamblarlo y luego se observará el impacto de pequeños cambios en el programa original (usando condicionales y nuevas operaciones). Se explicará el rol del compilador y se hará una revisión muy superficial del assembler para que se identifiquen las diferencias de longitud y sintaxis con el programa original. Se presentará el código binario y su relación con el assembler.
Herramientas
C y BARF

Clase 10 - Arquitectura básica de una computadora

Objetivos
Comprender en líneas generales las vinculaciones entre hardware y software a través de la relación entre los componentes con sus funciones.
Contenido y desarrollo
Arquitectura del computador a alto nivel. Reconocer que hay un procesador, memoria, periféricos y cables que los conectan. Reconocer las distintas partes (sobre todo periféricos) en distintas computadoras: de escritorio, netbook, teléfono celular, etc.
Herramientas

Partes de *hardware* en vivo o, en su defecto, video y/o fotos.

Clase 11 - Almacenamiento y memoria I

Objetivos
Aportar información y criterios de análisis para reconocer distintos tipos de almacenamiento.
Contenido y desarrollo
Clase 1 de 2 de Memoria. Comprender que hay una jerarquía y por qué existe (tiempo de acceso). Introducir terminología Byte, KB, MB, GB, TB, etc. Retomando la clase de variables, analizar la utilidad de la memoria e indagar en celulares y computadoras sobre los espacios de almacenamiento y el uso de recursos. Se compararán distintos tipos de archivos para presentar las unidades de medida y sus magnitudes y distintos programas para analizar el consumo de recursos.
Herramientas
Archivos y programas relevantes para el trabajo propuesto.

Clase 12 - Almacenamiento y memoria II

Objetivos
Analizar el funcionamiento de los distintos tipos de memoria.
Contenido y desarrollo
Clase 2 de 2 de Memoria. Comprender el concepto de caché y qué problema pretende resolver: el acceso rápido a la información frecuente. Esquematizar las relaciones entre los 2 niveles de Memoria y trabajar distintas ideas posibles para almacenar y liberar un dato de caché. Comprender el funcionamiento a alto nivel de un disco rígido y cuáles son sus partes constitutivas: discos, brazo lector, motor. Comparar las diferencias de tiempo de acceso entre caché, memoria RAM y disco rígido.

Herramientas
-

Clase 13 - CPU I

Objetivos
Comprender la relación entre las especificaciones de hardware y el funcionamiento de la CPU.
Contenido y desarrollo
Clase 1/2 de CPU. Qué hace la CPU a alto nivel (ejecutar una instrucción tras otra). Velocidad (terminología específica GHz). Especificaciones reales. 1 núcleo vs muchos núcleos. Se indagará sobre las especificaciones de los procesadores presentes en celulares y computadoras. Se analizarán las magnitudes en GHz para abordar su vinculación con las instrucciones vistas al desensamblar en la clase 9 y explicar por qué hay computadoras más rápidas que otras.
Herramientas
-

Clase 14 - CPU II

Objetivos
Entender en más detalle el funcionamiento de la CPU.
Contenido y desarrollo
Clase 2/2 de CPU. Cuál es el camino que realiza la CPU para ejecutar una instrucción. Activación de circuitos. Partiendo del análisis de la CPU de la clase anterior, se profundizará sobre cómo se elige la instrucción a ejecutar, cómo se realizan operaciones aritméticas y cómo se relacionan con los

espacios de almacenamiento que se trabajaron en clases pasadas. Finalmente, se retomará el concepto de “binario” para explicar que una computadora está constituída por la interconexión de muchos componentes electrónicos por los que circula electricidad.
Herramientas
-

Clase 15 - Relación hardware-software

Objetivos
Adquirir una perspectiva general del funcionamiento de una computadora que vincule hardware y software entre sí considerando todos los componentes trabajados anteriormente, la estructura general y sus partes e interrelaciones.
Contenido y desarrollo
Actividad de repaso o TP para reflexionar desde una perspectiva general la relación Alto Nivel-Bajo Nivel. Esta clase busca sintetizar los contenidos trabajados anteriormente. Partiendo desde el bajo nivel (circuito) hasta mayores niveles de abstracción, se presentará un Trabajo Práctico que proponga un análisis en camino inverso al recorrido realizado hasta ahora para abordar los temas. Propuesta: Repaso. Trabajo grupal en clase en el que se reparten varios programas y se pide una descripción de lo que ocurre durante su ejecución.
Herramientas
-

Clase 16 - Sistemas Operativos I

Objetivos

Descubrir que entre el hardware y las aplicaciones que usamos existe una capa intermedia: el Sistema Operativo.
Contenido y desarrollo
Retomar el programa C que los estudiantes hicieron en la clase 8 y compilarlo en dos sistemas operativos distintos. Comparar los dos binarios obtenidos y determinar que no son iguales a pesar de que ni el código del programa original ni la máquina que usaron cambiaron. A partir de esa observación se motivará una introducción a los sistemas operativos como mediadores entre la parte física de la computadora y el software que usamos. Se distinguirá entre las funciones básicas del Sistema Operativo, es decir, el kernel y la interfaz de usuario. Para ejemplificar, se mostrarán distintas interfaces de un mismo Sistema Operativo. Se introduce el trabajo práctico de la clase 19.
Herramientas
Huayra Linux

Clase 17 - Sistemas Operativos II

Objetivos
Comprender la utilidad y funcionalidades de los Sistemas Operativos y cuáles son sus partes más importantes.
Contenido y desarrollo
Se trabajará la idea de Sistema Operativo como un programa que funciona como nexo entre el hardware y el resto de los programas. Se distinguirá entre las funciones básicas del Sistema Operativo, es decir, el kernel y la interfaz de usuario. Para ejemplificar, se mostrarán distintas interfaces de un mismo Sistema Operativo. Se abordarán algunas nociones básicas de seguridad y se discutirá qué características o prestaciones de un Sistema Operativo analizar para poder decidir, entre distintas opciones, cuál se adecúa mejor a nuestras necesidades.

Herramientas
-

Clase 18 - Sistemas Operativos III

Objetivos
Comprender algunas características más avanzadas de los Sistemas Operativos.
Contenido y desarrollo
Se abordarán algunas nociones básicas de seguridad y se discutirá qué características o prestaciones de un Sistema Operativo analizar para poder decidir, entre distintas opciones, cuál se adecúa mejor a nuestras necesidades. Se trabajarán los conceptos de sistemas de paquetes y de sistema de permisos.
Herramientas
-

Clase 19 - Software libre y cultura libre

Objetivos
Entender las características del software libre y la cultura libre.
Contenido y desarrollo
Software libre, propietario y abierto. Comunidades. Derechos de propiedad. Búsqueda de información. TP de investigación.
Herramientas
-

Clase 20 - Software malicioso

Objetivos
Aportar información y criterios de análisis para reconocer el software malicioso, su origen y su funcionamiento. Favorecer el uso responsable de las tecnologías de la información y la comunicación.
Contenido y desarrollo
Virus. ¿Cómo protegerse? ¿Por qué son programas? Intercambio sobre distintos tipos de software malicioso. Ejemplos con sitios web con ads y pendrives. Descarga de ejecutables. Problemas de la autoejecución. Cómo evitar el phishing.
Herramientas
Caja de herramientas

Clase 21 - Repaso de programación

Objetivos
Repasar los conceptos de programación trabajados con anterioridad.
Contenido y desarrollo
Se programará un juego que combine los temas de programación vistos en las clases iniciales, con el objetivo de retomar la idea de programa entendido como guión, la modificación del orden secuencial de sus instrucciones a partir de alternativas condicionales y la interacción con el usuario en el transcurso de la ejecución de programa, que brinda información que será guardada en una variable en memoria para su uso posterior y permitirá trabajar con información desconocida y cambiante al momento de crear el programa.
Herramientas
Alice

Clase 22 - Métodos II

Objetivos
Profundizar la comprensión del uso de métodos como estrategia de abstracción.
Contenido y desarrollo
Se volverá a trabajar con la animación “Encuentro de tercer tipo”, pero bajo el formato de un juego del tipo *Elige tu propia aventura*. Para ello se eliminará el final de la historia y se lo dejará en suspenso a la espera de la decisión del usuario, a quien se le presentarán dos alternativas de desenlace. A partir del trabajo con un juego más complejo se buscará que los alumnos vuelvan sobre los conceptos anteriores, para recuperar su significado y apropiarse de los recursos aprendidos. Se trabajará nuevamente sobre la lectura de una historia por lo que se espera que se recupere nuevamente el proceso de trabajo para la creación de un programa: la lectura del guión, la identificación de métodos o acciones más abarcativas con los que organizar el accionar de los personajes, y paso siguiente la implementación del programa con Alice.
Herramientas
Alice

Clase 23 - Proyecto integrador I

Objetivos
Planificación inicial del proyecto de programación de un juego por parte de los alumnos. El proyecto grupal irá complejizándose a medida que se avance con los contenidos de programación en las sucesivas clases.
Contenido y desarrollo
Se delinearán las pautas iniciales para la programación de un juego en Alice. Se definirán la bases del juego que se irán complejizando con nuevas consignas y recursos a lo largo de las clases.

Herramientas
Alice

Clase 24 - Métodos III

Objetivos
Profundizar la estrategia de abstracción dada por los métodos, la pertinencia de su creación y la definición de un nombre adecuado.
Contenido y desarrollo
A partir de la presentación de un enunciado se les presentará a los alumnos divididos en grupos 6 programas como posibles soluciones. La propuesta es que los alumnos elijan la abstracción más adecuada que resuelva el enunciado. Se les pedirá que dentro de cada grupo debatan cuál de ellas es la solución más adecuada al problema.
Herramientas
Alice

Clase 25 - Repetición simple

Objetivos
Repaso del concepto de secuencialidad y de alternativas condicionales como dos tipos de *recorridos* de las instrucciones dentro de un programa. Introducción de estructuras de repetición simple, que modifican el flujo secuencial del programa y repiten las instrucciones una cantidad fija de veces.
Contenido y desarrollo
Se retoma la idea de secuencialidad de un programa y se vuelve a traer a escena la alternativa condicional desde esa perspectiva. Es importante resaltar la capacidad del condicional de

alterar el flujo secuencial de un programa, para que no transcurra de manera lineal. Se propone continuar rompiendo con la secuencialidad con otro tipo de recursos como la repetición simple.
Herramientas
Alice

Clase 26 - Variables II

Objetivos
Parte I: Por otro lado se profundizará el funcionamiento de las variables, con un uso no explorado: utilizarlas para contar una cantidad. Ello a partir de su uso como contador de puntaje dentro de la dinámica de un juego. Con esta actividad se espera también reforzar la idea de que una variable permite memorizar un valor que puede cambiar a lo largo de la ejecución del programa. ***Parte II:*** Se retomará el trabajo con el proyecto integrador. Como recurso de utilidad para el juego se verán brevemente los eventos en Alice que nos permiten evaluar respuestas del usuario (una tecla o el botón del mouse presionado). Se verán las múltiples posibilidades de Interactividad básica en Alice, para enriquecer el funcionamiento de los juegos. Avances con últimos detalles del proyecto integrador del juego.
Contenido y desarrollo
El objetivo es implementar un contador de puntaje para el jugador, que comience en cero y se incremente o decremente en función del accionar del jugador. Se profundizará el funcionamiento de las variables, con un uso no explorado: utilizarlas para contar una cantidad. Ello a partir de su uso como contador de puntaje dentro de la dinámica de un juego. Con esta actividad se espera también reforzar la idea de que una variable permite memorizar un valor que puede cambiar a lo largo de la ejecución del programa. Se avanzará con el proyecto integrador del Juego, definiendo los últimos detalles y resolviendo las dudas que existan.
Herramientas
Alice

Clase 27 - Repetición condicional

Objetivos
Introducir la repetición condicional (el “while” o “repetir mientras que…”), como una forma más avanzada de romper la secuencialidad de un programa. Comparar con el funcionamiento de la repetición simple, diferenciando a la repetición condicional por su evaluación de una condición para definir la cantidad de repeticiones. Destacar la potencialidad -dentro de la creación de un juego- de poder programar una repetición desconociendo la cantidad de repeticiones que se producirán.
Contenido y desarrollo
A la hora de introducir la repetición condicional, se trabajará sobre casos en los que la cantidad de repeticiones no son conocidas de antemano, para remarcar su diferencia con la repetición simple vista anteriormente. Este punto se aprovechará para explorar el azar en la definición de un juego. Por otro lado, se retomará la idea de condición, vista con la alternativa condicional que -en este caso- será la que indique en qué momento detener la repetición de las instrucciones contenidas dentro del bloque.
Herramientas
Alice

Clase 28 - Proyecto integrador II

Objetivos
Continuar con el trabajo en el proyecto final.
Contenido y desarrollo
En esta clase se continuará con el trabajo en el proyecto final, aprovechándose para resolver dudas y consultas, y monitorear el progreso.
Herramientas
Alice

Clase 30 - Proyecto integrador III

Objetivos
Presentación del proyecto final.
Contenido y desarrollo
En esta clase los distintos grupos mostrarán su proyecto final, presentándolo para el resto de la clase.
Herramientas
Alice

Secuencia de actividades

Clase 1 - Introducción

El docente hará su presentación y una breve descripción de la materia.

En caso de no tener Alice[3], se procederá a su instalación en todas las computadoras disponibles, mientras tanto, el docente puede realizar una pequeña encuesta para indagar sobre opiniones, relación con la tecnología, conocimientos previos y cualquier información que considere relevante, por ejemplo, para establecer canales de comunicación con los estudiantes.

Una vez concluída la instalación, se preguntará a los alumnos qué cosas creen que son necesarias para hacer una película. Se espera que hablen de luces, cámaras, personajes, escenografía y especialmente, que se mencione el concepto de **guión**. En caso de que no lo propongan, puede orientarse a los alumnos para que esta idea surja de ellos: ¿Y cómo saben los actores lo que tienen que hacer? ¿La película cuenta una historia? ¿Dónde está esa historia antes de que empiece la grabación?

Luego de ese intercambio, se les contará que Alice es una herramienta que sirve para hacer sus propias animaciones y juegos, se les preguntará por animaciones y juegos que conozcan y cómo y por quién piensan que están hechas.

Se comenzará a mostrar el entorno, en este punto es muy importante detallar cómo cambiar el idioma si no se encuentra en español.

Se agregarán personajes, se indicará cómo se ven en el árbol de objetos y aparecen en el escenario.

Una vez agregados uno o dos personajes, se preguntará qué piensan que va a pasar en la animación y se apretará el botón “Ejecutar”. Se indicará el lugar donde va el guión. La conclusión a la que se debería arribar es que si el guión está vacío, nada ocurrirá al comenzar la ejecución.

En este momento los alumnos podrán familiarizarse con la herramienta y explorar las galerías y características de la misma. Se pedirá a los estudiantes que hagan que dos personajes se muevan por el escenario.

Finalmente, se mostrará cómo hacer que un personaje haga algo, arrastrándolo desde el árbol de objetos al espacio del guión y eligiendo la acción deseada.

---

[3] Alice es una herramienta de programación para adolescentes que puede descargarse de http://www.alice.org/ de manera gratuita. Para facilitar su uso en las netbooks recomendamos descargar la versión 2.4. Se trata de la opción mencionada como "Current Release with complete Spanish Gallery- 2.4.3" en http://www.alice.org/index.php?page=downloads/download_alice2.4. La herramienta se descarga en inglés y luego mediante una opción se la pasa a nuestro idioma.

Clase 2 - Noción de programa y proyecto simple I

La clase comenzará con una propuesta simple en Alice: hacer una animación que incluya un personaje y un vehículo en la que el vehículo se aproxime a gran velocidad y cuando esté por chocar al personaje lo esquive y retome su camino. Además, se pedirá a los estudiantes que respondan algunas preguntas:

- ¿Qué ocurre si se intercambia el orden de las instrucciones?
- ¿Cuál es la diferencia entre arrastrar un método al espacio del guión con hacer click derecho en un personaje del árbol de objetos y seleccionarlo desde ahí?
- ¿Para qué usarían cada caso?

Estas preguntas serán el pie para explicar que el guión es un programa, ver que las instrucciones se ejecutan en orden, de a una por vez y distinguir entre el momento en que se escribe y el momento en que se ejecuta nuestro programa.

Adicionalmente se presentará un proyecto grupal en el que los alumnos deberán crear una animación o videojuego que aborde una temática (de su interés, a elección del docente o en el marco de algún proyecto que esté abordando la institución). El proyecto se hará fuera del horario de clase y se retomará más avanzado varias clases más adelante.

Durante el resto del trimestre se reservarán pequeños espacios de las clases para preguntar por el nivel de avance del proyecto, incentivar su continuación y resolver dudas. Para que el proyecto sirva como ejercitación de algunas de las temáticas de programación que se verán en la primera parte de la materia, será parte de la consigna que el proyecto incluya algún tipo de variabilidad que deberá poder determinarse en tiempo de ejecución. Por ejemplo, en base a una pregunta al usuario o botón a apretarse al comienzo, la historia podrá terminar bien o mal, transcurrir de día o de noche, etc.

Clase 3 - Métodos I

Se presentará el escenario “Encuentro de tercer tipo”[4] y se les propondrá a los alumnos que creen un programa a partir de un guión disponible en el archivo “Guión del encuentro de tercer tipo”. (Ambos archivos deberán ser compartidos a los alumnos a través de un pendrive o Internet). Para comenzar la actividad se leerla en conjunto y voz alta la historia:

Después de viajar por el espacio, una nave tripulada por un robot acaba de aterrizar en la Luna. El robot instala en la superficie una cámara que nos permite ver la escena: el robot, la nave espacial y la Luna. Lo primero que se observa es al robot emitiendo un sonido de exploración. De repente un extraterrestre aparece de atrás de unas rocas, y dice "Bienvenidx". El robot encara al alien, pega un salto y gira su cabeza de la sorpresa, vuelve a emitir el mismo sonido, se acerca al extraterrestre para inspeccionarlo, luego mira hacia la cámara, vuelve a emitir el sonido de exploración, la cabeza del robot se pone roja y dice: "¡Houston, tenemos un problema!".

Y se les propondrá a los alumnos que comiencen a programar el guión.

A medida que trabajen con la historia, se observará que la secuencia de acciones de los personajes -una atrás de la otra- se vuelve extensa y complicada de leer.

Luego de que hayan avanzado con el programa, se interrumpe la clase, y se les dice a los alumnos que el guión cambió y se les indica que ahora el robot primero inspecciona al alien y luego se sorprende.

Se deja que los alumnos cambien su programa a partir de la nueva consigna.

Transcurrido un tiempo se detendrá la actividad y se volverá sobre la historia para trabajar en conjunto con las siguientes preguntas disparadoras:

- ¿Cómo podríamos haber escrito el programa para simplificar los cambios necesarios al modificar la historia?
- ¿Cómo podemos subdividir el comportamiento del robot en acciones más abarcativas?

Se espera que los alumnos identifiquen acciones más generales que involucren varios movimientos del robot. Por ejemplo, es posible identificar que el robot -de acuerdo al guión inicial- se sorprende, investiga y reacciona. Otras respuestas también pueden ser correctas.

Se propondrá buscar -por ejemplo- la acción "investigar" dentro de las acciones que puede realizar el personaje (solapa métodos). Como no se encontrará la instrucción investigar, se volverá a preguntar: ¿Qué movimientos tiene que llevar a cabo el robot para “Investigar” al alien recién descubierto de acuerdo al guión?

La propuesta será entonces modificar los programas para incluir estas acciones más generales bajo la forma de métodos. Es decir, programar a partir de la identificación de acciones de un mayor nivel de abstracción que involucren una serie de pasos más simples y cuenten con un fin específico.

[4] Actividad basada en “Learning to program with Alice” de Wanda Dann, Stephen Cooper y Randy Pausch.

Se les pedirá a los alumnos que armen un método para cada acción identificada, con un nombre ilustrativo y que lo incluyan en sus programas. Y una vez definidos los métodos para que el robot cumpla con la totalidad del guión, se ejecutará la animación.

En muchos casos es probable que se programen las instrucciones dentro de los métodos pero sin invocarlos, por lo que al ejecutar la animación el robot se quedará quieto. En otros casos es posible que se invoquen los métodos nuevos pero no se incluya ninguna instrucción dentro de ellos, por lo que el robot también se quedará quieto.

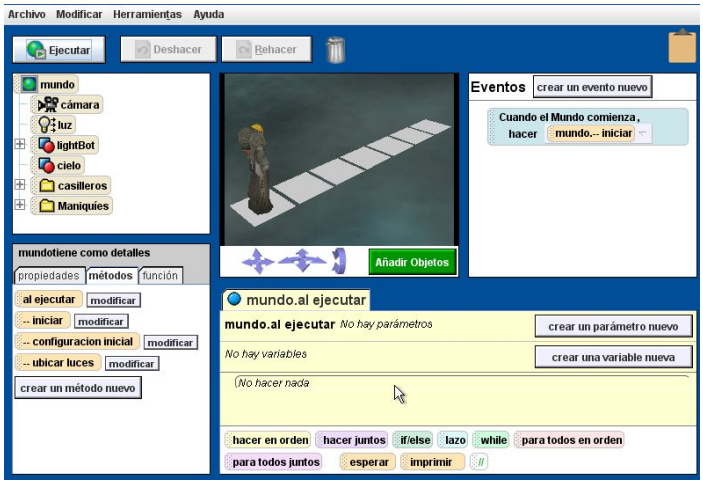
Ante consultas de este tipo es importante indicar que en la estrategia de creación de un método son necesarias dos instancias. Por un lado especificar las instrucciones concretas que componen esa acción más abstracta. Y por otro llamar o invocar al método nuevo en el lugar adecuado dentro de nuestro programa.

Para cerrar, vale la pena plantear el hecho de que los programas que realizamos no sólo se deben hacer entendibles para la computadora sino que también es importante que otra persona pueda leer nuestro programa y entenderlo. (Se puede traer a colación la estrategia de introducir comentarios en el código).

Clase 4 - Alternativa condicional

Se repasará el concepto de método utilizando el escenario inicial disponible en Lightbot.a2w[5], que los alumnos deberán abrir en Alice.

Luego de cargar el archivo se obtiene el siguiente escenario inicial:



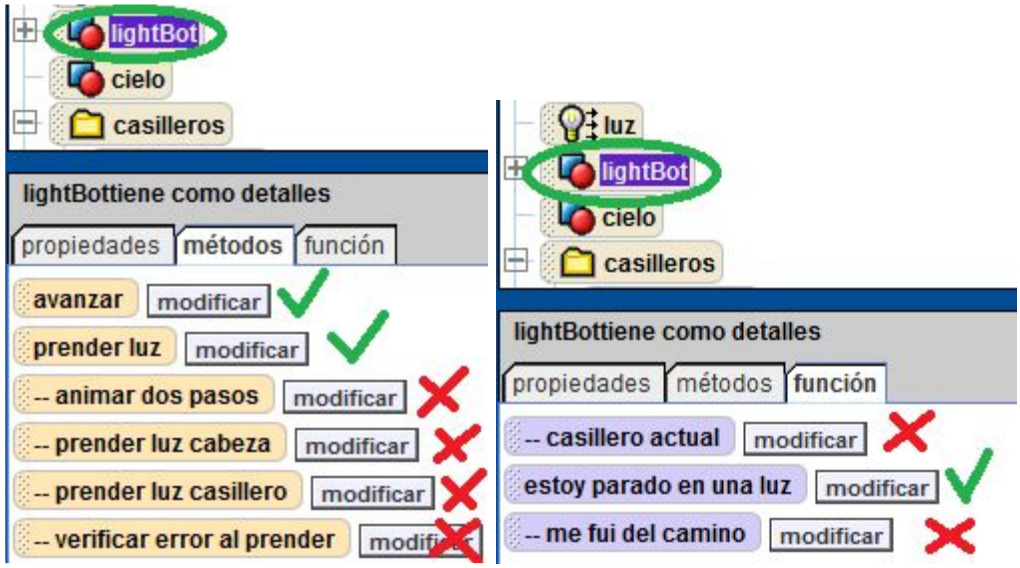
Escenario inicial de Alice con la actividad LightBot.a2w

Una vez hecho esto, se les pedirá a los alumnos que ejecuten reiteradas veces el programa, y que tomen nota de los cambios que se producen en cada ejecución. A partir de este escenario inicial los alumnos deberán programar al robot Lightbot para que avance de a un casillero por vez. En cada casillero Lightbot deberá ver si la luz está prendida o apagada, y en el caso de que esté apagada prenderla antes de continuar.

Antes de comenzar, se repasará en conjunto con los alumnos el concepto de método visto la clase anterior y se señalarán los métodos disponibles (“avanzar”, “prender luz” y “estoy parado en una luz”) para cumplir la consigna. Para ello se deberá tener en cuenta que únicamente deberán usarse aquellos métodos que no comienzan con “--”. Estos últimos son parte de la configuración inicial, por lo que no será necesario utilizarlos en la resolución del problema.

A continuación se muestra los métodos que pueden usarse:

[5] Esta actividad está inspirada en los juegos educativos de Lightbot. Más información en https://lightbot.com/



Los métodos tildados con verde son los que pueden usarse para resolver el problema. Los que empiezan con un doble guión "--" son parte de la configuración inicial.

Se insistirá en el funcionamiento cambiante del escenario en cada ejecución y se les pedirá a los alumnos que hagan un programa que resuelva la consigna.

Luego que realicen varios intentos para resolver el ejercicio, es probable que los alumnos encuentren dificultades a la hora de definir cuándo encender o no la luz, dado que las luces a encender cambian en cada ejecución. Se presentarán las alternativas condicionales como la forma en que la computadora evalúa la información que tiene en ese momento de la ejecución y toma un camino en consecuencia. En este caso, nos permitirá evaluar si la luz de un casillero se encuentra encendida o apagada.

Se pedirá a los estudiantes ejemplos de la vida cotidiana (Si llueve llevo paraguas, Si tengo hambre me como un alfajor, etc.) y ejemplos en juegos o aplicaciones que conozcan (Si Mario toma el hongo cambia de tamaño, si me disparan pierdo puntaje, si pongo la contraseña incorrecta no puedo entrar a Facebook, etc.)

Se presentará la instrucción if/else de Alice explicando la traducción "Si… entonces…" y se propondrá terminar el ejercicio. A partir de la instrucción if/else es posible evaluar -para cada casillero- si el personaje está parado en una luz, y sólo en ese caso encenderla y avanzar. Se insistirá a los alumnos acerca de la pertinencia de usar un método nuevo para llevar a cabo esa evaluación, que podrá llamarse "prender luz si hay".

Una solución posible para el problema es la siguiente:

Para finalizar, se mostrarán brevemente los eventos del teclado para que puedan explorar opciones de interactividad y puedan programar juegos interactivo. Hasta este punto hemos construido animaciones con los que el usuario no interactúa. Es decir, al ejecutar nuestros programas observamos con exactitud la sucesión de instrucciones que hemos programado. Los eventos nos dan la posibilidad de que el usuario interactúe con el programa y que las acciones de los personajes respondan a, por ejemplo, clics del mouse o teclas presionadas por el usuario. Para ilustrar este punto se les mostrará a los alumnos una modificación al programa anterior con el uso de eventos del teclado. Para convertir nuestra animación en un juego, haremos que cuando el usuario presione la flecha derecha del teclado Lightbot avance un casillero. Es decir, la presión de una tecla como evento provocará como respuesta la ejecución del método avanzar. Y, como segundo evento, cuando el usuario presione la barra espaciadora el personaje se deberá fijar si hay o no una luz en ese casillero.

Se avisará a los alumnos que deberán traer el avance del proyecto presentado en la clase 3 para la clase siguiente.

Clase 5 - Proyecto simple II

En esta clase se trabajará sobre el proyecto presentado en la clase 2. Cada grupo expondrá su idea original, el nivel de avance y si encontraron nuevas herramientas o alguna limitación al momento de programar su pieza. Se propondrán ideas y objetivos a futuro y se realizará un intercambio para abordar de manera conjunta los aspectos de la herramienta que requiera cada pieza.

Clase 6 - Variables I

Se preguntará la diferencia entre juegos y animaciones. El objetivo es que los alumnos se aproximen a la idea de interactividad. Se retomará la diferencia entre tiempo de escritura y tiempo de ejecución visto en la clase 2 y se les mostrará cómo hacer que el usuario ingrese su nombre durante el transcurso del programa y algún personaje lo diga.

Como ejercicio para introducir variables, se pedirá que en un mundo con 3 personajes hagan que ingresando una única vez el nombre del usuario todos los personajes lo saluden.

Los estudiantes se encontrarán con la dificultad de no poder recuperar el nombre ya que no fue guardado en ningún lado. Esto nos dará una buena oportunidad para presentar el concepto de variable y hacer una primera aproximación a la idea de memoria y almacenamiento.

Para ilustrar la dificultad que encontraron para recuperar el nombre, puede realizarse una representación con la colaboración de 3 alumnos:

1. Se le dará un papel al primer alumno para que anote un nombre.
2. Se pedirá al segundo alumno que lo lea.
3. Se tirará el papel.
4. Se pedirá al tercer alumno que también lea el nombre, pero ya no tendrá de donde hacerlo.

Luego de repetir esta situación dos veces, se hará lo mismo pero incorporando una caja o sobre que será una representación de la variable:

1. Se le dará un papel al primer alumno para que anote un nombre.
2. Se guardará el papel en la caja/sobre.
3. Se pedirá al segundo alumno que lo lea.
4. Se guardará el papel en la caja/sobre.
5. Se pedirá al tercer alumno que lea el nombre.

Se discutirá con el curso cómo creen que debería llamarse la caja/sobre: ¿con el nombre que anotó el primer alumno? ¿Qué pasa si se repite la secuencia pero preguntando el nombre al segundo alumno?

Se recreará otra situación en la que se me modifica el nombre que se guarda en la caja:

1. Se le dará un papel al primer alumno para que anote un nombre.
2. Se guardará el papel en la caja/sobre.
3. Se pedirá al segundo alumno que lo lea.
4. Se guardará el papel en la caja/sobre.
5. Se pedirá al tercer alumno que anote un nombre.
6. Se guardará el papel en la caja/sobre en lugar del otro
7. Se pedirá al primer alumno que lo lea.

Con esta dramatización deberá quedar en claro que denominar a la caja con alguno de los nombres en particular no tiene sentido y se escribirá “Nombre” en ella. Luego se explicará que la

caja representa una variable, que es un espacio de memoria donde se almacena información para poder recuperarla o modificarla y la importancia de que se la nombre con claridad (en función del tipo de contenido que tendrá y no con un contenido en particular).

Se mostrará como crear y utilizar variables en Alice y se invitará a los alumnos a terminar el ejercicio propuesto.

Clase 7 - Diferencia entre programa y algoritmo

Se comenzará recapitulando brevemente los conceptos de programación que se vieron hasta la clase pasada: métodos, alternativa condicional y variables. Este repaso resulta conveniente puesto que todas las actividades requerirán utilizar estos conceptos para su resolución.

A continuación se propondrá realizar en parejas el siguiente ejercicio en Alice:

- Un personaje pregunta al usuario por el resultado de un juego entre dos equipos en el que pueda haber un ganador o empate (fútbol, básquet, handball, rugby, etc.). El usuario deberá primero ingresar el puntaje del primer equipo y luego le puntaje del segundo. El personaje de Alice debe decidir si ganó el primer equipo, el segundo o hubo empate.

Para poder resolver este ejercicio será necesario utilizar interactividad (“preguntar al usuario un número”), variables (para guardar ambos puntajes) y, al menos, dos alternativas condicionales (if/else) para determinar si hubo un ganador o hubo empate. Notar que solamente con una única alternativa condicional no sería posible discernir entre 3 resultados posibles.

A medida que el docente vaya recorriendo las mesas de trabajo y notando que todos los grupos pudieron avanzar en la actividad, se propondrá una puesta en común en donde se analicen distintas propuestas de solución (dos if/else anidados, 3 if/else secuenciales, etc.) y si la siguiente es una solución válida:

Figura 7.1

Algunas preguntas que pueden orientar la discusión del programa de la figura 7.1:

- ¿Podemos saber si el programa funciona bien sin ejecutarlo en Alice?
- ¿Qué posibles resultados deberíamos considerar para realizar un análisis?
- Si en el segundo condicional reemplazáramos el “>” por un “≥”, ¿qué pasaría?

- Si el programa tuviera muchas más líneas y variables, ¿creen que sería fácil detectar un problema de este tipo?

Aquí se puede hacer una breve digresión acerca de que a este tipo de defectos se los conoce como bugs (*bichos* en inglés)[6] y que al programar, como en cualquier actividad humana, suelen cometerse errores que generan defectos que hacen que nuestro programa no se comporte de la manera deseada.

Una buena práctica, que resulta de utilidad antes de comenzar a programar, es diseñar un algoritmo que resuelva el problema. ¿Qué es un algoritmo? Hay muchas definiciones distintas pero simplemente diremos que **un algoritmo describe una estrategia de resolución de un problema reconociendo cuáles son sus partes centrales**. Para definir la estrategia se puede (y sugerimos) utilizar el castellano con oraciones que describan un comportamiento específico.

Se puede pensar entre todos cuál sería un algoritmo para el ejercicio anterior. Daremos el siguiente a modo de ejemplo:

```
Preguntar al usuario los resultados de A y B
Si A es más grande que B
        Ganó A
Si A es más chico que B
        Ganó B
Si A y B son iguales
        Empataron
```

A este lenguaje a medio camino entre el castellano que usamos cotidianamente y un programa en un lenguaje de programación lo llamaremos *pseudocódigo*. Diseñar una estrategia de resolución o algoritmo, nos permite pensar de manera más abstracta cuáles serían las grandes acciones que deberíamos realizar para resolver el problema. Con el algoritmo diseñado resulta más sencillo pasar luego a un programa.

A continuación se les propondrá a las y los estudiantes que diseñen un algoritmo para resolver el siguiente problema:

- Se dispondrán 4 personajes en el escenario. Uno de ellos les preguntará la edad a los otros 3 y determinará quién es el más grande. Las edades las deberá ingresar el usuario a medida que se vayan preguntando, es decir, no están fijadas en el código del programa. Notar que debe decir el nombre del personaje de mayor edad y no cuál de las edades es la más grande.

El/la docente deberá prestar atención y enfatizar que la idea no es programar la solución en Alice ni escribirla en papel con las mismas instrucciones que Alice provee. A medida que vayan realizando los algoritmos, el/la docente puede motivar a las y los estudiantes a que chequeen si su algoritmo funcionaría para distintos casos: que el primero sea el más grande, el segundo o el tercero. Si hubiera dos personajes con la misma edad, ¿qué haría su algoritmo?

[6] https://es.wikipedia.org/wiki/Error_de_software#Or.C3.ADgenes_del_t.C3.A9rmino

Luego de que lo hayan terminado, se realizará una puesta en común y se analizarán distintas propuestas de algoritmos:

- ¿Todos devuelven un nombre?
- ¿Todos funcionan?
- En el caso de que hubiera edades repetidas, ¿todos devuelven el mismo nombre?

Es interesante ver que para el mismo problema hay distintos abordajes o estrategias de resolución, notando que, por ende, también habrá muchos programas que lo resuelvan (si los algoritmos propuestos por las y los estudiantes fueran similares, el/la docente puede proponer uno diferente y sumarlo al análisis).

Como cierre, el/la docente comentará que los algoritmos no se restringen sólo a la programación sino que en nuestra vida cotidiana también los usamos y diseñamos:

- Cuando dividimos dos números **usamos** el algoritmo de la división.
- Cuando damos las indicaciones para llegar de un punto a otro **diseñamos** un algoritmo que indica cuáles son los pasos que debemos realizar para llegar a destino.
- Si para cocinar **usamos** una receta de cocina, estamos siguiendo los pasos definidos en ella para obtener el plato que queremos, es decir, un algoritmo culinario.

Por lo tanto, es importante diferenciar entre algoritmo y programa: mientras el primero determina una serie de pasos para resolver un problema de manera automática y utiliza el lenguaje castellano, el segundo también determina una serie de pasos o instrucciones pero éstas se ejecutan en una computadora y se definen en un lenguaje de programación como Alice, es decir, un lenguaje que la máquina conoce.

Clase 8 - Lenguajes de programación

Durante esta clase se usarán herramientas que ya vienen instaladas en Huayra y no vienen por defecto en Windows. Por este motivo, para no dedicar tiempo a la instalación de estas herramientas, sugerimos usar durante toda la clase el Sistema Operativo Huayra. Cada estudiante deberá tener una carpeta en el directorio "alumno" o en el "Escritorio"[7] con los siguientes archivos: `compilar_y_ejecutar.sh` , `funciones.h`, `funciones.c` y `entrada_de_cine.c`. Los archivos se encuentran disponibles en el siguiente link: https://goo.gl/hqZRj0. Para descargar todos los archivos, al abrir el link, arriba a la derecha aparecerá una flecha similar a  . Al hacer clic allí, se descargará el archivo "`TI3_clase_8_archivos.zip` ", que contiene comprimidos los 4 archivos mecionados. Para extraerlos, cliquear en el archivo con el botón derecho del mouse y luego en "Extraer aquí".

Para poder ejecutar el archivo "`compilar_y_ejecutar.sh` " haciendo doble clic en él es necesario agregarle permisos de ejecución[8]. Para ello, en cada compu, hay que hacer clic derecho en el archivo "`compilar_y_ejecutar.sh`", ir a "Propiedades", luego "Permisos"[9] y, por último, hacer clic en "Permitir ejecutar el archivo como un programa".

Al comenzar, se recordará brevemente lo que hicieron la clase pasada: dado un problema, aprendieron a diseñar un algoritmo en un lenguaje denominado pseudocódigo.

Como primera actividad, se pensará entre todos un algoritmo que resuelva el siguiente problema:

- Dory comenzó hace pocos días a trabajar como vendedora de entradas en el viejo cine "El Cambalache submarino". Como tiene especialmente poca memoria, quiere hacer un programa que le diga, dada la edad de una persona, cuál es el valor de su entrada. Para los que tienen 5 años o menos y para los que tengan 60 años o más, la entrada cuesta 20 caracolas. Para el resto, la entrada tiene un valor de 50 caracolas.

Un algoritmo posible es

```
Preguntarle la edad al usuario
Si tiene 5 años o menos
        Paga 20
Si tiene 60 años o más
        Paga 20
En cualquier otro caso
        Paga 50
```

Una vez generado consenso acerca de cuál es el algoritmo que resuelve el problema de Dory, se propondrá programarlo. Pero esta vez no será en Alice sino en otro lenguaje de programación llamado C. A diferencia de Alice, en C las instrucciones del programa hay que escribirlas en modo texto en un archivo que se conoce como "fuente".

---

[7] No guardar en la carpeta "Documentos" ya que no se pueden cambiar los permisos del script "`compilar_y_ejecutar.sh`".
[8] Wiki sobre permisos de archivos en Huayra: http://wiki.huayra.conectarigualdad.gob.ar/index.php/Permisos
[9] En las clases 17 y 18, en donde se trabajará sobre Sistemas Operativos, se dará una noción de qué son los permisos y para qué sirven.

En esta clase (y en este curso) se utilizará una parte mínima del lenguaje C, de manera muy simplificada[10]. Para introducir a las y los estudiantes en la sintaxis del lenguaje y cómo ejecutar los programas, este primer problema lo programarán entre todos junto con el/la docente. Una buena manera para empezar es observar qué es lo primero que hay que hacer según el algoritmo que escribieron recién. Si lo tuvieran que programar en Alice, ¿qué harían primero? Probablemente haya consenso en que habría que "preguntar al usuario un número" para conocer cuál es su edad. En la versión simplificada del lenguaje que usaremos, pedirle al usuario que "ingrese su edad", en C lo podemos hacer del siguiente modo:

```
imprimir_un_texto("Ingrese su edad:");
int edad = leer_entero();
```

`imprimir_un_texto()` es una instrucción que imprime por pantalla solamente texto, el cual se escribe entre comillas dobles.

En C, al igual que en Alice, cuando queremos crear y usar una variable, debemos indicar si se trata de un número, de un texto, o de algún otro tipo de valores. En Alice, cuando creábamos la variable se nos ofrecía una lista con los posibles tipos[11] que podía tener. En C, para crear una variable indicamos primero su tipo -**`int`**- y luego su nombre, **`edad`** en este caso. Además, al crearla, podemos asignarle un valor. Por ejemplo, la edad que ingresa el usuario mediante la instrucción **`leer_entero()`**. Si lo quisiéramos hacer en dos pasos, primero crear la variable y luego asignarle un valor, el código quedaría del siguiente modo:

```
imprimir_un_texto("Ingrese su edad:");
int edad;
edad = leer_entero();
```

Cabe destacar que en C todas las instrucciones terminan con un punto y coma. ¿En Allice cómo se separaban las instrucciones? Otro aspecto que se observa en estas pocas instrucciones es que la función que imprime un texto tiene entre paréntesis las palabras que van a ser impresas pero, en cambio, la que lee la edad que ingresa el usuario no tiene nada entre paréntesis. Esto se debe a que la instrucción que imprime necesita conocer el valor que va a escribir en la pantalla pero la instrucción que lee la edad del usuario no necesita ningún valor previo para poder ejecutarse, simplemente se fija qué valor ingreso el usuario y lo devuelve.

Al finalizar esta construcción colectiva de las primeras líneas de código, se les puede pedir a las y los estudiantes que escriban estas instrucciones en su archivo "`entrada_de_cine.c`". Las mismas deben ir debajo de la línea que dice "`// ESCRIBIR EL CÓDIGO AQUÍ`". El archivo se puede abrir en cualquier editor de texto que tengan instalado.

[10] Para no abordar en esta instancia algunos aspectos engorrosos de la sintaxis de C, como puede ser leer un valor que ingresa el usuario o imprimir un texto por pantalla, en el archivo "`funciones.c`" ya están programadas algunas funciones que nos van a permitir realizar estas operaciones de manera más sencilla.
[11] Recordemos que el *tipo* de una variable indica qué valores puede almacenar y qué operaciones se pueden hacer entre ellos. Por ejemplo, las variables enteras o *int* almacenan números mientras que las de tipo texto o *string* almacenan texto diverso.

En este punto, se puede preguntar a la clase qué creen que va a ocurrir si ejecutan este programa dado que no tenemos personajes ni escenario. Para correr el programa, hay que hacer doble clic en el archivo "`compilar_y_ejecutar.sh`". Este archivo es un programa muy simple que recibe el archivo con el programa que hicieron, lo traduce a código de máquina para que lo podamos ejecutar[12] y lo ejecuta.

Lo que verán al ejecutar el programa será una pantalla negra, llamada *terminal*, en donde aparecerá el texto "Ingrese su edad:". El programa habrá ejecutado la primera de las instrucciones de nuestro programa y se encuentra esperando que el usuario ingrese un valor para la edad. Similar a lo que ocurría en Alice cuando aparecía un pop-up en donde ingresábamos un valor. Luego de ingresar un número y presionar "Enter", el programa terminará y no hará más nada, debiendo cerrar la ventana de la terminal a mano.

Para verificar que el número que ingresamos haya sido guardado correctamente en la variable, podemos agregar la instrucción **`imprimir_un_numero(edad)`**, obteniendo el siguiente programa:

```
imprimir_un_texto("Ingrese su edad:");
int edad = leer_entero();
imprimir_un_numero(edad);
```

A continuación se propone seguir construyendo entre todos el programa en C que resuelve el problema según el algoritmo que plantearon hasta llegar a tener un programa similar a:

```
imprimir_un_texto("Ingrese su edad:");
int edad = leer_entero();

int valor_entrada;
if (edad < 6) {
      valor_entrada = 20;
} else {
      ...
}
...
```

En los 3 puntos suspensivos las y los estudiantes deberían terminar de escribir el programa, para lo cual ya tienen todas las herramientas de sintaxis de C necesarias.

[12] Se profundizará este aspecto en la siguiente clase.

Un posible programa al que podrían arribar y que resuelve el problema es

```
imprimir_un_texto("Ingrese su edad:");
int edad = leer_entero();

int valor_entrada;
if (edad < 6) {
      valor_entrada = 20;
} else {
      if (edad > 59) {
            valor_entrada = 20;
      } else {
            valor_entrada = 50;
      }
}

imprimir_un_texto("El valor de su entrada es");
imprimir_un_numero(valor_entrada);
```

Se puede establecer la similitud entre la instrucción **`if/else`** que usaban en Alice y la instrucción **`if {} else {}`**, las cuales son equivalentes. Simplemente cambia la forma en que se escriben. Mientras en Alice para el **`if`** y para el **`else`** tenemos un lugar especial donde encastrar las instrucciones que queremos ejecutar en cada caso, en C ese espacio lo delimitamos con las llaves. Notar que el segundo **`if {} else {}`** está dentro del primer **`else {}`**. Esto es así puesto que si la persona no tuviera menos de 6 años -la condición del primer **`if {}`**- todavía no podemos saber si tiene más o menos de 60 años. Por ende, en el caso de que tuviera 6 años o más y entrásemos en el primer **`else {}`**, deberíamos preguntar si tiene 60 años o más para poder terminar de decidir si tiene que abonar una entrada de 20 caracolas o de 50.

A continuación, se les pedirá a las y los estudiantes que, de la carpeta en la que están trabajando, borren el archivo "`entrada_de_cine`", sin confundir con el archivo "`entrada_de_cine.c`". Luego, hacer doble clic en "`compilar_y_ejecutar.sh`" y ver qué ocurre en esa misma carpeta. Lo que pasará es que se creará nuevamente el archivo "`entrada_de_cine`". ¿Por qué ocurre esto? ¿Qué hace ese nuevo archivo?

Como microactividad opcional, si hay tiempo, se los puede invitar a que lo abran con un editor de textos y ver si pueden inferir algo de su contenido. Como se trata de un archivo binario, todo lo que verán será una secuencia de caracteres que aparentemente no tiene sentido. ¿Qué significa todo este texto incomprensible? ¿Qué pasaría si agregamos una línea al final del código y volvemos a hacer doble clic en "`compilar_y_ejecutar.sh`"? ¿Cambia este archivo? Para que puedan comparar si hubo cambios en ese archivo antes y después de cambiar una línea de código, se sugiere que antes de volver a compilar y ejecutar, cambien el nombre del archivo por "`entrada_de_cine_anterior`" o alguna variante similar que dé cuenta que se trataba de la primera versión del archivo. Luego de realizar algún pequeño cambio en el código del programa, por ejemplo, agregar una instrucción que imprima un texto al final, compilar y ejecutar, se generará un nuevo archivo "`entrada_de_cine`". Si queremos comparar caracter a caracter este archivo con el anterior, va a resultar difícil y tedioso. La manera más sencilla es comparar los

tamaños de los archivos. Si uno fuera más grande que otro, entonces no podrían contener el mismo texto. De hecho, el más grande será "`entrada_de_cine`". ¿Por qué ocurre esto? ¿Cuál será la relación entre el código del programa que hicieron y este archivo? Aparentemente si tenemos más líneas de código parecería que el archivo generado es más grande. Estos temas se profudizarán en la clase siguiente.

Para terminar, el/la docente contará que además de Alice y C, existen muchos otros lenguajes de programación, y la elección de uno u otro depende de varios motivos: el contexto del problema, el lenguaje de moda en ese momento, el gusto del programador, etc. Sin embargo, con cualquier lenguaje de programación, ya sea visual como Alice o textual como C, podemos programar cualquier algoritmo.

Créditos

Autores

(por orden alfabético)
Teresa Alberto
Fernando Schapachnik
Herman Schinca
Daniela Villani

Diseño gráfico e ilustración

Jaqueline Schaab

Coordinación Iniciativa Program.AR

María Belén Bonello
Pablo Factorovich
Fernando Schapachnik
www.program.ar

Autoridades Fundación Dr. Manuel Sadosky

Presidente: Dr. Lino Barañao
Director Ejecutivo: Dr. Esteban Feuerstein
www.fundacionsadosky.org.ar