Tema 2. Introducción a la programación mediante programación por bloques

Este módulo de programación por bloques tiene como objetivo introducir a los alumnos en la utilización de metodologías y herramientas accesibles para aprender conceptos de programación mediante actividades entretenidas y amenas, buscando fortalecer el pensamiento computacional vinculado a la resolución de problemas.

Se utilizará la herramienta de programación PilasBloques Las actividades están basadas en el curso de Introducción a la Programación, elaborado por la Fundación Sadosky. (www.fundaciónsadosky.com).

Martínez López (2013) presenta un enfoque novedoso para la enseñanza de la programación guiado por la necesidad de focalizar el aprendizaje en el proceso de abstracción, y en los conceptos fundamentales, transversales a todos los paradigmas y lenguajes, y describe un marco conceptual que incluye dos grupos de conceptos técnicos: las herramientas conceptuales y las del lenguaje, que se detallan en la tabla 1.

Las **herramientas conceptuales** son tres: 1) la noción de estrategia de solución y su explicitación en división en tareas o bloques; 2) la noción de que los programas son un medio de comunicación entre personas, por lo que es importante que sean legibles, lo que conlleva la elección de nombres adecuados para los bloques o módulos de un programa; 3) la noción de algorítmica básica, expresada en la idea de recorrido y control de la secuencia.

Las **herramientas del lenguaje,** incorporadas en los lenguajes de programación, incluyen: 1) comandos para describir acciones (primitivas, procedimientos, alternativa, repetición), y 2) expresiones, para describir datos (parámetros, variables, datos primitivos).

Tabla 1: Marco conceptual para la enseñanza de la programación

Herramientas conceptuales	**Estrategia de solución y división en subtareas o módulos**
	Legibilidad: elección de nombres adecuados
	Algorítmica básica: recorridos
Herramientas del lenguaje	**Acciones** (comandos - “verbos”) - Comandos primitivos; secuencia de comandos - Procedimientos y parámetros - Repetición simple - Alternativa condicional - Repetición condicional
	Datos (expresiones - “sustantivos”) - Valores (literales numéricos y otros) - Sensores y datos primitivos; sensores de interactividad - Operadores, parámetros y Variables

1. Herramientas conceptuales

a) La noción de *estrategia de solución*, y la de su explicitación para aplicar la metodología de **división en subtareas o módulos**,

b) La noción de que los programas son fundamentalmente un medio de comunicación entre personas, además de servir como vehículo para hacer funcionar máquinas, capturado en la importancia de que los programas sean **legibles** (o sea, claramente entendibles por otros programadores) y esto a su vez expresado mediante la metodología de *elección de nombres adecuados* para cada una de las partes de un programa que se escribe.

c) La noción de algorítmica básica, expresada en este curso en forma simplificada, por ejemplo, en la noción de recorrido.

a) Estrategia de solución y división en subtareas o módulos

Toda vez que se busca solucionar algún problema, es necesario contar primero con alguna idea de cómo encarar dicha solución, o sea, qué elementos disponer para la solución y de qué manera.

Esto en programación se conoce como **estrategia de solución**: ¿qué cosas considerar a la hora de expresar/proponer la solución? ¿Cuáles son los componentes que interactuarán en la solución para obtener la respuesta deseada (lograr el objetivo)

Una forma de expresar estrategias de solución es considerando pequeños problemas cuyas soluciones combinadas provean la solución al problema general.

Esta forma de descomponer un problema en sub-problemas (o una tarea a realizar en sub-tareas) es una de las bases conceptuales de la programación. La **división en subtareas o módulos** representa a la forma de pensar composicionalmente y es una herramienta conceptual invalorable en el pensamiento de alto orden.

Vale la pena remarcar que mientras que la estrategia de solución es una idea particular de QUE hacer para resolver un problema, la división en subtareas es una forma específica de CÓMO expresar dicha estrategia a través de soluciones a problemas más pequeños.

Los ejercicios que se realizaran en la práctica, si bien son muy sencillos, son básicos para el objetivo de este tema: explicitar la estrategia de solución elegida expresada a través de subtareas, y luego expresar cómo se lleva a cabo cada subtarea. Esto se logra combinando las herramientas del lenguaje con las demás herramientas conceptuales de manera adecuada; pero el concepto fundamental subyacente en todo momento es **explicitar** la estrategia de solución a través de la adecuada división en subtareas (además de la habilidad de nombrar adecuadamente cada una y de utilizar las herramientas del lenguaje adecuadas para construir la solución).

La herramienta del lenguaje básica para expresar subtareas y estrategias es el **procedimiento.**

¿Qué es un procedimiento?

Es un conjunto de primitivas expresadas en el orden adecuado para crear una nueva tarea. Es útil para "enseñar" una nueva tarea al autómata.
En un procedimiento, además de primitivas, pueden utilizarse también otros procedimientos ya definidos.
Conviene definir un procedimiento cuando identificamos un "patrón".
Un patrón es algo que ocurre con regularidad.
Una Regularidad es todo aquello que se presenta en un orden periódico, es decir, tiene una sucesión regular. Se trata de una característica que se puede apreciar en los acontecimientos, en una sucesión numérica o en cualquier sucesión temporal en la que sea apreciable una cierta ordenación de las cosas.

Una regularidad puede darse tanto en un conjunto de datos como en acciones que se deben llevar a cabo. Las regularidades están en todas partes y ser capaz de encontrarlas es una habilidad esencial.

La ventaja de detectar un patrón de acciones que el autómata debe realizar es que se puede "enseñar" una sola vez al autómata a realizar esta nueva tarea, mediante un procedimiento (secuencia ordenada de acciones), y luego se puede volver a utilizar esta tarea las veces que sean necesarias, sin necesidad de volver a escribir toda la secuencia de acciones.

Resumiendo, los pasos recomendados para solucionar un problema son:

1. Idear una estrategia de solución, y explicitarla;
2. Expresar la estrategia mediante una división en bloques o módulos;
3. Declarar y nombrar adecuadamente cada bloque de modo que exprese la tarea que realiza;
4. Definir cada uno de los bloques que expresan tareas, mediante instrucciones primitivas

b) Legibilidad

Para entender el concepto de legibilidad, primero tenemos que saber que "un programa es una descripción ejecutable de la solución a un problema computacional".

Los programas son entidades "duales" dado que deben indicarle a una máquina cómo funcionar, pero, también deben comunicar la solución propuesta a las personas.

O sea, los programas son manipulados por dos entidades: **máquinas**, pero también **personas**. Las definiciones y propuestas tradicionales ponen todo el énfasis en la máquina que ejecuta el programa y en su funcionamiento, y dejan en segundo plano el valor comunicacional del programa, invisibilizando de alguna manera a las personas que los programan.

En esta asignatura se enfatiza el valor comunicacional de un programa y se valora la "legibildad" de las soluciones propuestas.

Se dice que un programa es **legible** cuando puede ser leído con sencillez por una persona, y entendido con poca o ninguna explicación adicional.

La legibilidad se ve expresada en la elección de nombres adecuados para las entidades que se escriban en el programa: procedimientos, parámetros, variables y otras entidades.

Si bien no existe una definición precisa de qué constituye un nombre adecuado, existen algunos criterios generales: Por ejemplo, es universalmente aceptado que los nombres de una o dos letras son extremadamente inadecuados en la mayoría de los casos, y también que los nombres extremadamente largos (por ejemplo, que ocupan todo un renglón o más), también. Se suelen favorecer nombres cortos pero descriptivos. En el caso de los comandos, por ejemplo, estos nombres usualmente requieren verbos en infinitivos: Avanzar, tocar, …

Adicionalmente, se requiere que el nombre brinde una idea correcta respecto del propósito de la entidad nombrada, pues si no usualmente conduce a confusiones que limitan o complican la legibilidad

La correcta utilización de estas herramientas conceptuales tiene un impacto directo en la **calidad de los programas que se escriben**. Esto es fundamental cuando se forman programadores profesionales, pues código de calidad significa mayor productividad. Sin embargo, esta forma de pensar y organizarse tiene valor más allá de las aplicaciones profesionales de la programación, y es por eso una de las razones por las que desde la iniciativa Program.AR, que provee el contexto de estas actividddaes, se busca generalizar la enseñanza de las Ciencias de la Computación.

Las estrategias de solución, la correcta comunicación de soluciones a problemas, y mecanismos elementales de solución de problemas sencillos habilitan y fomentan la formación de capacidades de pensamiento de alto orden, que hoy día es aceptado como forma fundamental del pensamiento humano.

c) Algorítmica básica: recorridos

Básicamente, un **recorrido** es un esquema de repetición en el cual se realiza una tarea para cada uno de los elementos de una serie, pues recorre la secuencia de elementos de a uno, procesando de alguna forma cada uno de ellos.

La idea de un recorrido es organizar la división en subtareas a la hora de resolver una tarea que requiere trabajar con cada uno de los elementos de una secuencia, de manera tal de simplificar la confección de una solución y garantizar el correcto funcionamiento de la misma. De esta forma, habrá subtareas para **iniciar** el recorrido, **determinar si se terminaron** los elementos, **procesar** un elemento, **pasar al siguiente elemento** y para **finalizar el recorrido**.

La secuencia de elementos que un recorrido se encarga de procesar puede ser muy concreta (por ejemplo, una serie de elementos en fila, como ser frutas o señales) o ser más abstracta (por ejemplo, una serie de posiciones distantes a visitar en un mapa o incluso un recorrido sobre números para resolver un problema numérico tal como calcular una factorial). Sin embargo, cuanto más abstracta resulte la secuencia, más complejo será transmitir la idea adecuadamente.

2. Herramientas del lenguaje

2.1. Acciones (comandos - "verbos")

2.1.1. Comandos primitivos; secuencia de comandos

2.1.2. Procedimientos y parámetros

2.1.3. Repetición simple

2.1.4. Alternativa condicional

2.1.5. repetición condicional

2.2. Datos (expresiones - "sustantivos")

2.2.1. Valores (literales numéricos y otros)

2.2.2. Sensores y datos primitivos; sensores de interactividad

2.2.3. Operadores, parámetros y Variables

En la programación, la forma de expresar las ideas y soluciones propuestas es, invariablemente, a través de un **lenguaje de programación**.

Un lenguaje de programación determinado establece cómo escribir programas en dicho lenguaje, y provee diferentes herramientas, y diferentes formas de combinarlas, pero todas esas formas se pueden entender como casos particulares de una serie genérica de herramientas.

Las herramientas del lenguaje se pueden clasificar según el tipo de elementos que describen: los **comandos** se utilizan para describir acciones y las **expresiones** se utilizan para describir datos.

En un lenguaje natural, las acciones se describen mediante verbos y los datos se describen mediante sustantivos. Por esa razón, se puede establecer la analogía y pensar a los comandos como "verbos" y a las expresiones como "sustantivos".

El programa se arma combinando verbos y sustantivos para brindar a la máquina que lo ejecutará la descripción adecuada para que pueda llevar a cabo la solución. Los comandos le indicarán acciones que debe realizar, y las expresiones le indicarán con qué datos deben llevarse a cabo las acciones. Ejemplo: Avanzar (10 pasos), el comando Avanzar indica la acción y la expresión (10 pasos), indica cuánto avanzar.

2.1. Acciones (comandos-verbos)

Los comandos se pueden clasificar de diversas maneras.

2.1.1. Comandos primitivos:

Descripción de las acciones elementales que la máquina que ejecutará el programa puede realizar; cada máquina considerada tendrá diferentes capacidades, lo cual se verá reflejado en el conjunto de comandos primitivos que se manejen desde el lenguaje.

La primera forma de combinación de comandos primitivos (y luego, de comandos en general) es la *secuenciación*. En una secuencia de 2 comandos, por ejemplo, se realiza la acción indicada por el primer comando y habiendo terminado esa acción, se realiza la acción indicada por el segundo comando. Es la forma básica de combinación, que resulta imprescindible para construir acciones más complejas.

2.1.2. Procedimientos:

La siguiente forma de combinación de comandos primitivos que se propone trabajar es el **procedimiento**. Los procedimientos son la herramienta que plasma de manera más precisa la noción de abstracción que se busca transmitir, y por eso se consideran una herramienta fundamental.

Un procedimiento en su forma más simple no es más que una acción compleja (obtenida, por ejemplo, por secuenciación de acciones más elementales) a las que se les asigna un nombre que la identifica. Puede entenderse esta herramienta también como una forma de definición de nuevos comandos. Por ejemplo, si se cuenta con el comando primitivo AvanzarUnPaso(), se puede definir el procedimiento AvanzarTresPasos() como la secuencia AvanzarUnPaso(); AvanzarUnPaso(); AvanzarUnPaso(). Tal cual se describió anteriormente, la elección de un nombre adecuado que describa de manera precisa el propósito del procedimiento es básica para la correcta legibilidad del programa.

Los procedimientos constituyen uno de los pilares fundamentales de este enfoque de enseñanza de la programación. Son la forma de explicitar en el lenguaje la división en subtareas, y por ello comparten la importancia de dicha herramienta conceptual.

La **división en subtareas** y **procedimientos** pueden verse como dos instancias del mismo concepto fundamental: **la abstracción**

- una del lado conceptual
- la otra del lado del lenguaje

Es importante hacer una observación sobre la denominación de **procedimiento,** en algunas herramientas o enfoques lo que aquí se denomina procedimiento se puede encontrar nombrado como funciones, bloques, módulos, subrutinas, o alguna otra forma. Pero en cada caso, es claro que se trata de un mecanismo de abstracción sobre comandos, una forma de definir nuevos comandos.

En su forma más simple, los procedimientos solamente le dan nombre a alguna acción compleja devenida a partir de combinación de otros comandos. Sin embargo, el verdadero poder expresivo de los procedimientos se encuentra cuando se los combina con la noción de **parámetro.**

Los parámetros son una herramienta que permite expresar muchos procedimientos diferentes mediante un único procedimiento parametrizado. Un parámetro es como un "agujero" que el procedimiento tiene, y que debe ser completado con un dato: el **argumento**. El argumento provisto para completar el parámetro determina cuál de las instancias específicas expresadas por el procedimiento parametrizado se busca utilizar.

Los parámetros son otra forma de conseguir abstracción: muchos casos particulares son expresados de una sola vez a través de un procedimiento parametrizado. Sin embargo, esta forma de abstracción es mucho más difícil de transmitir que la de procedimiento, especialmente a

estudiantes que recién se inician en la programación. Por esa razón, en este curso los parámetros aparecen de manera tardía, y no se hace un fuerte hincapié en ellos.

La noción de **parámetro y parametrización** es crítica en la formación de un programador profesional.

2.1.3. Repetición simple y condicional

La clasificación de comandos continúa con diversas formas de combinación de comandos: dos formas de repetición (simple y condicional) y una forma de alternativa (condicional). Estas herramientas permiten aumentar el poder expresivo de la persona que programa al habilitar o facilitar comportamientos complejos, que son o bien difíciles o bien imposibles de expresar utilizando solamente las herramientas previas.

A la hora de presentar repeticiones es conceptualmente más simple comenzar con una forma de repetición que repite un número fijo de veces.

La forma avanzada de repetición que se presenta en el curso es la que se denomina repetición condicional, pues la cantidad de repeticiones no se conoce a priori, y se utiliza una condición como mecanismo para determinar cuándo dejar de repetir. Esta forma de repetición es conocida por muchos con nombres menos descriptivos, como “while” o “do while” (mientras) o “repeat until” (repetir hasta); sin embargo, es conveniente utilizar la denominación de repetición condicional, pues es más descriptiva y adecuada a la hora de ubicarla en el marco conceptual.

2.1.4. Alternativa condicional

La alternativa condicional es una forma de elegir entre dos posibles cursos de acción, basándose en una condición. Es comúnmente conocida como “**if-then-else**” o simplemente “**if**” o a veces “condicional”. Pero estas formas familiares de denominarla ignoran el hecho fundamental de que lo que se está expresando es una alternativa entre dos posibles acciones.

2.2. Expresiones

La siguiente categoría de elementos de un lenguaje de programación que consideramos es la de expresiones, que son la manera de describir información, y por ello equiparables a los “sustantivos” de un lenguaje natural.

En su forma más básica, las expresiones aparecen en forma literal, por ejemplo, para representar números (2, 17, 42, etc.) o cadenas de caracteres ("hola", "Este es un string", etc.) o valores de verdad (verdadero y falso).

Las expresiones pueden ser:

- nombres que representan información de manera indirecta: parámetros y variables, y
- sensores y datos primitivos que representan la información que la máquina puede recolectar del medio ambiente en el que ejecuta.

Se retomará el tratamiento de expresiones en el tema 4.

Método de resolución de problema

El principal objetivo de esta introducción a la programación con actividades lúdicas es incorporar los conceptos básicos de la programación (herramientas del lenguaje) y consolidar el pensamiento computacional que se requiere para la programación (herramientas conceptuales).

En base a estos conceptos, se propone un método de resolución de problema que sirva para asimilar con mayor facilidad la tarea de programar y facilitar la transición hacia la programación “convencional” mediante la aplicación (transversal) de este método, cuyos pasos se detallan en la Tabla 2.

Tabla 2. Método de resolución de problemas

Pasos
– Analizar el problema (Ver el escenario) – Determinar el QUE (objetivo) – Pensar una solución global
Pensar un nombre para la solución
– Detectar "**patrones**". Esto es reconocer situaciones que se repiten y se pueden **reutilizar**. (En algunos casos pueden no presentarse patrones).
– Pensar una **estrategia** – Dividir el problema en **tareas** (procedimientos)
Pensar **nombres representativos** para las tareas
Escribir la solución global
- Analizar las primitivas - Determinar el COMO - Armar la solución final con: ▪ las primitivas ▪ los procedimientos ▪ las repeticiones ➔ **CODIFICAR**
VERIFICAR: ejecutar y comprobar que se haya logrado el objetivo establecido en el primer paso. Esto implica conocer de antemano el **resultado esperado**.