

Tema 1. Introducción a la programación

1. Conceptos fundamentales

La programación es un campo de estudio propio de la disciplina Informática, por ello es importante conocer la Informática o Ciencias de la Computación como área de conocimiento que abarca un conjunto de conocimientos específicos.

Informática o Ciencias de la Computación

Por Informática se entiende al “Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático y racional de la información por medio de computadoras”. Justamente la palabra informática, de origen francés, está formada por la contracción de los vocablos: **INFORMación** y **autoMÁTICA**.

En esta definición se destacan dos términos relevantes:

- **Tratamiento automático:** Se refiere a que son las propias máquinas (las computadoras) las que realizan las tareas de captura, proceso y presentación de la información.
- **Tratamiento racional:** Se refiere a que todo el proceso está regulado a través de una secuencia de instrucciones (programa) que siguen el razonamiento humano.

El objetivo marcado es el **tratamiento automático** de la información y el medio utilizado para lograrlo es **la computadora**.

Por lo tanto, para entender en qué consiste la Informática, es imprescindible conocer:

- Todas las características técnicas de las computadoras y sus componentes asociados (periféricos).
- Los distintos tipos de información y datos que se manejarán (sistemas de representación, archivos y bases de datos).
- Los procesos y métodos aplicados a la información o datos (programas).
- Los sistemas de comunicación que permitirán tratar la información a distancia y compartirla de forma confiable (redes de comunicación de datos).

Conocer con detalle estos aspectos y la relación que existe entre ellos, permitirá entender cómo se realiza el tratamiento automático de la información utilizando computadoras.

En la definición se hace mención a la utilización de “computadoras”. En la práctica, el término es más amplio y debe entenderse como “sistema informático”, siendo este un conjunto de recursos destinados al tratamiento de la información.

La Informática como disciplina

La Association for Computing Machinery (ACM) (1) elaboró una serie de recomendaciones sobre la Currícula de Computación, y en este documento especifican la siguiente definición:

La Computación como disciplina comprende el estudio de procesos algorítmicos que describen y transforman a la información; estudian su teoría, análisis, diseño, eficiencia, implementación y aplicación. La cuestión fundamental es: ¿qué puede ser automatizado (en forma eficiente)? (2)

La disciplina Informática o Ciencias de la Computación, nació en la primera parte de la década de 1940 con la conjunción de la teoría de los algoritmos, lógica matemática y la invención de la computadora electrónica con programa almacenado.

Dentro de la disciplina se identifican distintas subáreas de conocimiento que se adecuan permanentemente en función de la evolución de la disciplina, algunas subáreas típicas son las siguientes:

- Estructuras de Datos y Algoritmos
- Lenguajes de Programación
- Arquitectura
- Computación Numérica y Simbólica
- Sistemas Operativos
- Ingeniería de Software
- Bases de Datos y Búsqueda de Información
- Inteligencia Artificial y Robótica
- Comunicación Humano-Computadora

La Informática como Ciencia y como Tecnología

Por sus características, la Informática puede concebirse como ciencia teórica y como ciencia de carácter empírico sobre los fenómenos relacionados con la información y la computación (3).

La estructura disciplinar de la Informática está formada por varias teorías (Teoría General de los Sistemas, Cibernética, Teoría de la Información y la Comunicación, Teoría de Lenguajes Formales y Autómatas, teorías matemáticas y lógicas) y su teoría central es la Teoría de la Computabilidad, vinculada con el concepto de algoritmo. Esta teoría explora la naturaleza de los problemas que son accesibles a la mente humana y los clasifica en los demostrablemente irresolubles y los resolubles que admiten un algoritmo para su solución.

La tecnología es una actividad social centrada en el “saber hacer” que, mediante el uso racional, organizado, planificado y creativo de los recursos materiales y la información propios de un grupo humano, en una cierta época, brinda respuestas a las demandas sociales en lo que respecta a la producción, distribución y uso de bienes, procesos y servicios.

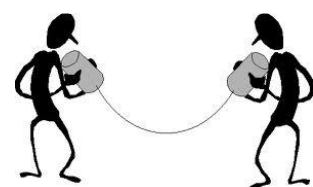
La Informática, como disciplina tecnológica, analiza determinados problemas relacionados generalmente con la adquisición, almacenamiento, procesamiento y transferencia de datos-información-conocimientos que plantea la sociedad y trata de buscar su solución relacionando la técnica (conocimientos, herramientas, capacidad inventiva) con la ciencia y con la estructura económica y socio-cultural del medio.

Ejemplos de tecnologías: Internet, plataformas educativas, teléfonos inteligentes.

Información y dato

La Informática tiene como objetivo principal proporcionar información en forma automatizada, por lo tanto, es necesario tener claro qué se entiende por INFORMACIÓN.

Por una parte, se puede denominar información a la “comunicación o adquisición de conocimientos que permiten ampliar o precisar los que se poseen sobre una materia determinada”. Por ejemplo, porcentaje de estudiantes que aprobaron la materia Álgebra.



Por otra parte, algunos autores denominan información al “conjunto de símbolos con los que se representan convencionalmente hechos, objetos e ideas”. Según este enfoque cualquier conjunto de símbolos es información, aunque no tenga sentido. Por ejemplo, un archivo de datos de estudiantes almacenado en el disco rígido.

Atendiendo a estas dos formas de concebir la información, desde la Informática, resulta más pertinente considerar a la información como una realización del conocimiento, conocimiento

como objeto, aunque unido a los símbolos del mensaje que soporta la información. Información como conocimiento derivado del análisis o procesamiento de los datos.

De esta manera, se puede entender que las computadoras tratan solo con la parte física de los símbolos: la representación de la información. Por ello, cuando se dice que una computadora procesa información se debe entender que manipula las cadenas de símbolos que portan la información. Es decir, la computadora recibe como entrada una representación simbólica de ciertos datos de un problema y proporcionará como producto cierta secuencia de símbolos, que, convenientemente interpretada, dará la solución (información) del problema. A continuación, se ilustra la idea con el gráfico que se muestra en la Figura 1.

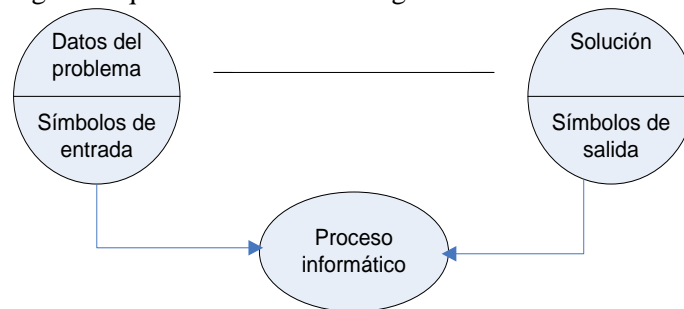


Figura 1. Proceso informático

En el contexto genérico de procesamiento de información, se denomina **dato** a la información de entrada e **información** a la salida o resultado del procesamiento.

Por ejemplo, un programa puede ingresar los datos de los estudiantes de la FaCENA: Apellido y Nombre, Sexo, Carrera, Materia que cursa y, mediante un proceso, seleccionar y mostrar cuáles son los estudiantes que cursan la asignatura Algoritmos y Estructuras de Datos I. Este resultado constituye la información resultante de este proceso particular. A su vez, esta información puede ser entrada de otro proceso que calcule e informe, por ejemplo, cuántas mujeres y cuántos varones cursan la asignatura Algoritmos y Estructuras de Datos I. ¿Cuáles serían en este último ejemplo los datos de entrada y cuál la información resultante?

Los datos de entrada serían los datos de los estudiantes que cursan Algoritmos y Estructuras de Datos I y los datos de salida serán cantidad de mujeres y cantidad de varones.

Entonces, el término “información” en Informática puede entenderse, en un sentido amplio, como el conjunto de símbolos con los que se representan convencionalmente hechos, objetos e ideas, y en una acepción más específica, **información como resultado del procesamiento realizado por una computadora**.

En tanto que, un dato es una representación de un objeto del mundo real mediante la cual se puede modelar aspectos de un problema que se desea resolver con un programa en una computadora.

A su vez, un dato está formado por uno o varios caracteres. Un **carácter** es cualquier símbolo numérico, alfabético o especial que se emplea en la escritura y en el cálculo, tal como:

- Numéricos: 0, 1, 2, ..., 9
- Alfabéticos: a, b, c, ..., z, A, B, C, ... Z
- Especiales: *, /, +, #, ...
- De control: salto de línea, fin de archivo (EOF)...

Estos símbolos, como veremos más adelante, se representan en la computadora como un conjunto de bits denominado **byte**. Entonces, cuando se refiere a caracteres almacenados, un byte es equivalente a un carácter.

Tratamiento de la información

Tratamiento de información quiere decir **operar** o **procesar** un conjunto de datos iniciales o datos de entrada, y, como resultado de este procesamiento, obtener un conjunto de datos finales o de salida. El procesamiento de datos está constituido por tres actividades básicas:

- a) **Captura de datos de entrada:** Los datos deben ser representados en el interior de la computadora antes de su proceso. Los datos pueden ser captados directamente por la computadora (por ejemplo, detectando electrónicamente un sonido, una temperatura, un código de barras) o pueden ser ingresados en forma de letras o números (caracteres).
- b) **Proceso de los datos:** Se realizan las operaciones necesarias para obtener, a partir de los datos de entrada, la salida o resultado esperado.
- c) **Manejo de los resultados de salida:** Los resultados ser almacenados, visualizados en pantalla o papel o se transferidos de un lugar a otro, mediante redes.

¿Por qué se automatiza el tratamiento de la información? Las razones que han llevado a la automatización del tratamiento de la información son fundamentalmente cuatro:

- La realización de funciones que el hombre por sí solo no puede llevar a cabo, por ejemplo, comunicaciones a larga distancia, etc.
- La ejecución de funciones que, aunque el hombre pueda realizarlas por sí mismo, su ejecución demandaría mucho tiempo. Por ejemplo, los cálculos complejos para el seguimiento y control de un proyectil dirigido o de una nave espacial.
- La obtención de seguridad en algunas tareas, como las que implican la repetición de una serie de pasos, en las que el hombre es más propenso a cometer errores.
- La sustitución del hombre para tareas monótonas. Ejemplo: control automático en la fabricación de productos industriales.

Necesidad de información

El conocimiento o información obtenida en el procesamiento se utiliza para **tomar decisiones** con vistas a un accionar concreto. Esta es la importancia que tiene la Informática en la actualidad: **permite obtener información confiable, precisa y oportuna que permitirá tomar mejores decisiones**. Esto posibilita a las empresas y organizaciones el logro eficiente de sus objetivos.

La Información, en el sentido de **conocimiento derivado del análisis o tratamiento de los datos**, permite **tomar decisiones** con vistas a un accionar concreto. Esta es la importancia que tiene la Informática en la actualidad, **permite obtener información confiable, precisa y oportuna que permitirá tomar mejores decisiones**. Esto posibilita a las empresas y organizaciones el logro eficiente de sus objetivos.

Ejemplo: El sitio institucional de la UNNE¹ brinda las estadísticas de los aspirantes a ingresar en las distintas Facultades. La figura 2 y 3 muestran la distribución de los estudiantes aspirantes en el año 2021 por Facultades y por género. Cómo aporta esta información a las decisiones que se tomen para promover la formación de determinados perfiles profesionales? O la mayor participación de las mujeres en carreras vinculadas con la ingeniería y la ciencia?

¹ <http://www.unne.edu.ar/homepage/unne-en-cifras/aspirantes>



Figura 2: Aspirantes por facultades

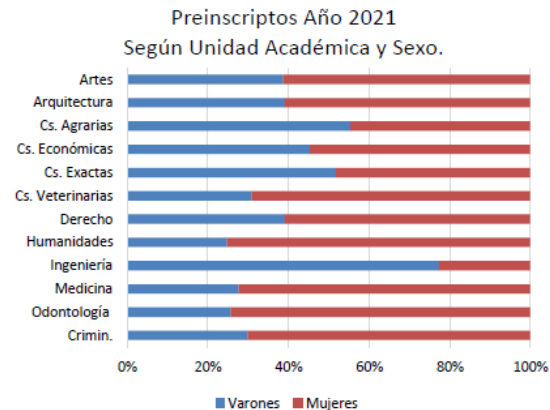


Figura 3: Aspirantes por facultades y género

Las organizaciones modernas manejan en la actualidad una gran cantidad de datos, producto de la sistematización de sus tareas operativas y de la interacción con otras organizaciones a través de las redes. La disponibilidad de información oportuna y confiable es fundamental para la supervivencia de las mismas.

Codificación de la información

En Informática es frecuente codificar la información. **Codificación** es una transformación que representa los elementos de un conjunto mediante los de otro, de forma tal que a cada elemento del primer conjunto le corresponda un elemento distinto del segundo.

Ejemplos de códigos: número de matrícula de un auto, código postal, ISBN de un libro, código de enfermedad definido por la Organización Mundial de la Salud.

Con los códigos se puede **comprimir** y **estructurar** la información. Por ejemplo: código de sexo: 1-Varón, 2-Mujer. Se almacena solamente el número asociado a la descripción.

Concepto de BIT y BYTE

En el mundo de las computadoras la información es **digital**, es decir, que está representada por **bits**. Un **bit** o dígito binario, es la unidad más pequeña de información que puede procesar una computadora. Puede tener uno de dos valores: **0** ó **1**. Representa la información correspondiente a la ocurrencia de un suceso de entre dos posibilidades distintas: prendido o apagado, abierto o cerrado, blanco o negro.

Si imaginamos el interior de una computadora como una colección de microscópicos conmutadores ON/OFF, como si fueran lámparas eléctricas, se puede por qué la información se procesa bit a bit. Cada conmutador o lámpara, puede estar prendido o apagado, indicando dos posibles estados: 0 o 1, como se muestra en la Figura 4.



Figura 4. Concepto de bit

Se utiliza la codificación para asignar valores a los estados: el valor numérico "0" si la lámpara está apagada y "1" cuando está encendida. Este código sencillo permite transmitir información. Por ejemplo, si establecemos que una lámpara prendida en la puerta de una habitación significa

"no se puede entrar" estaremos recibiendo un mensaje perfectamente definido. Una sola lámpara permite codificar dos estados distintos (por ejemplo: "se puede entrar" y "no se puede entrar").

Pero si tenemos dos lámparas podríamos codificar cuatro ($4=2^2$) estados distintos, considerando las posibles combinaciones de prendido y apagado:

- a) las dos lámparas están apagadas (0,0)
- b) la primera está encendida y la segunda apagada (1,0)
- c) la primera está apagada y la segunda encendida (0,1)
- d) la dos están encendidas (1,1)

De esta forma una computadora puede procesar fragmentos más grandes de información tratando grupos de bits como **unidades lógicas**. Por ejemplo, una colección de 8 bits, llamada **byte**, puede representar 256 mensajes diferentes ($256=2^8$). Esta codificación permite representar, por ejemplo, las letras del alfabeto, los dígitos decimales y símbolos especiales.

La Figura 5 muestra la codificación de la letra "A" que corresponde al decimal 65 y que internamente se representa con 8 bits, el primero y el último prendido y los restantes apagados.

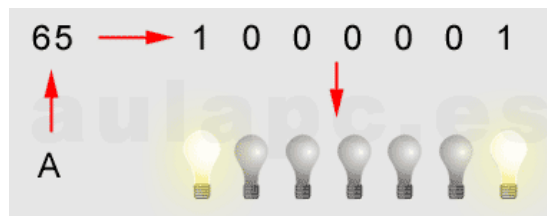


Figura 5. Representación digital de la letra A



Para mejorar la comprensión de estos conceptos, mirar el video de Adrián Paenza: *Alterados por pi - Sistema de numeración binario*:
<https://www.youtube.com/watch?v=iJkXq9kmQnc>

Qué representan los bits

En la Figura 6 se ilustra como el texto, los números y los símbolos (caracteres especiales) se representan en el interior de la computadora como unos y ceros, y se muestran fuera de la computadora en el formato entendible por las personas.

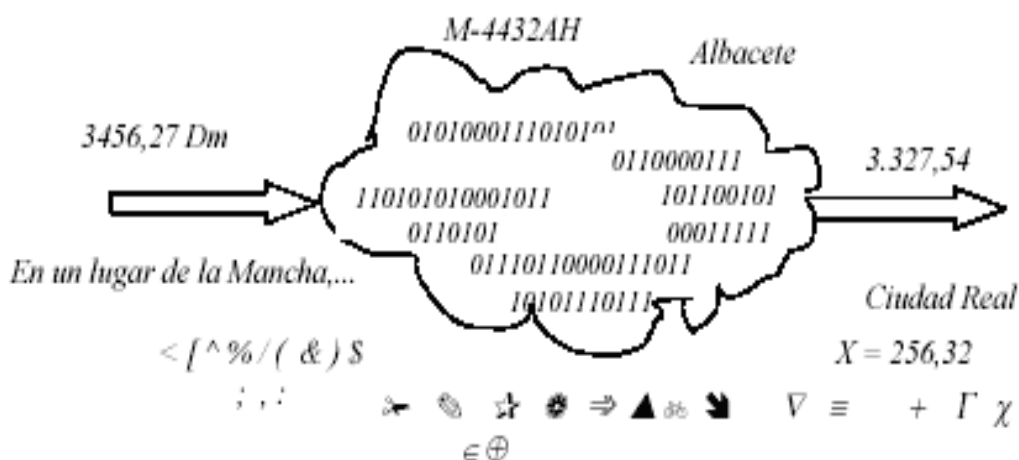


Figura 6. Representación interna de la información

Unidades de medida para el almacenamiento de información

Para medir longitudes se utilizan los metros, para medir capacidad se usan los litros, para medir el peso se utilizan los gramos y el tiempo se mide en horas, minutos y segundos. Para medir la capacidad de almacenamiento de información, se utilizan los bytes.

Dentro de la computadora la información se almacena y se transmite en base a un código que sólo usa dos símbolos, el 0 y el 1, y a este código se le denomina **código binario**.

Las computadoras reducen toda la información a ceros y unos, es decir que representan todos los datos, procesos e información con el *código binario*, un sistema que denota todos los números con combinaciones de 2 dígitos, el 0 y el 1. Es decir que el potencial de la computadora se basa en sólo dos estados electrónicos: encendido y apagado. Las características físicas de la computadora permiten que se combinen estos dos estados electrónicos para representar letras, números y colores.

Entonces, la unidad más elemental de información es un valor binario conocido como **bit**, es una posición o variable que toma el valor 0 ó 1. Representa la información correspondiente a la ocurrencia de un suceso de entre dos posibilidades distintas: prendido o apagado, abierto o cerrado.

En Informática, cada letra, número o símbolo ocupa un byte (8 bits). Por ejemplo, si un archivo de texto ocupa 5.000 bytes decimos que equivale a 5.000 caracteres. Un **byte** es el número de bits necesarios para almacenar un carácter.

La capacidad de almacenamiento de una computadora o de un soporte de información (tal como el disco rígido, un CD o un pen drive) se mide en *bytes*. Como es una unidad relativamente pequeña, es usual utilizar múltiplos, que se muestran en la Tabla 1.

Las denominaciones de las unidades derivan del **Sistema Internacional de Unidades** (http://es.wikipedia.org/wiki/Sistema_Internacional_de_Unidades).

Tabla 1: Unidades de medida de la información almacenada

MEDIDA	EQUIVALENCIA	EJEMPLOS
1 Byte	8 bits	<i>Una letra, un número o un símbolo</i>
1 Kilobyte (KB)	1024 bytes = 2^{10} bytes	<i>Un archivo de texto, 20 KB</i>
1 Megabyte (MB)	1024 KB = 2^{20} bytes = 1.048.576 bytes	<i>Un archivo mp3, 3 MB</i>
1 Gigabyte (GB)	1024 MB = 2^{30} bytes = 1.073.741.824 bytes	<i>una película en DivX, 1 GB</i>
1 Terabyte (TB)	1024 GB = 2^{40} bytes	<i>800 películas, 1 TB</i>
1 Petabyte (PB)	1024 TB = 2^{50} bytes	<i>En diciembre de 2007, YouTube tuvo un tráfico de 27 PB</i>
1 Exabyte (EB)	1024 PB = 2^{60} bytes	<i>La información almacenada en el mundo (al 2011) es de aproximadamente 300 EB</i>
1 Zettabyte (ZB)	1024 EB = 2^{70} bytes	<i>No existe ejemplo real</i>
1 YottaBytes	1024 ZB = 2^{80} bytes	<i>No existe ejemplo real</i>

Se utiliza el factor multiplicador 1024 en lugar de 1000, por ser la potencia de 2 más próxima a 1000.

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.fundacionsadosky.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 sub tareas o módulos
	Legibilidad: elección de nombres adecuados
	Algorítmica básica: recorridos
Herramientas del lenguaje	Acciones (comandos - “verbos”) <ul style="list-style-type: none"> - Comandos primitivos; secuencia de comandos - Procedimientos y parámetros - Repetición simple - Alternativa condicional - Repetición condicional
	Datos (expresiones - “sustantivos”) <ul style="list-style-type: none"> - 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 invaluable 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 sub-tarea. 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 “legibilidad” 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 actividades, 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 cuál 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 3.

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
<ul style="list-style-type: none"> – Analizar el problema (Ver el escenario) – Determinar el QUE (objetivo) – Pensar una solución global
Pensar un nombre para la solución
<ul style="list-style-type: none"> – Detectar “patrones”. Esto es reconocer situaciones que se repiten y se pueden reutilizar. (En algunos casos pueden no presentarse patrones).
<ul style="list-style-type: none"> – Pensar una estrategia – Dividir el problema en tareas (procedimientos)
Pensar nombres representativos para las tareas
Escribir la solución global
<ul style="list-style-type: none"> - Analizar las primitivas - Determinar el COMO - Armar la solución final con: <ul style="list-style-type: none"> ▪ las primitivas ▪ los procedimientos ▪ las repeticiones <p>➔ CODIFICAR</p>
VERIFICAR: ejecutar y comprobar que se haya logrado el objetivo establecido en el primer paso. Esto implica conocer de antemano el resultado esperado .