

Qué es programar y diferentes formas de programación

Teoría O.A.

Cosas administrativas



ILyPC (Introducción a la Lógica y los Problemas Computacionales)

Es la primera materia de las carreras de informática.

Algunas características (las expandimos más adelante):

- Presencial de lunes a jueves (según comisión),
- Asistencia obligatoria (75% de las clases mínimo **incluye virtuales**)
- 2 exámenes (instancia parcial, cada una con su correspondiente recuperatorio)

Se dicta en múltiples días y horarios:

- Hay en total 26 comisiones.
- Todas tienen 4 horas de duración, dos horas presencial y dos virtuales.
- Docentes turno mañana Alejandra, Silvia, Fabiana Patricia L, Patricia C, Alberto, Luisa y Viviana
- Docentes turno tarde: Fernando, Inés, Viviana y Patricia
- Docentes turno noche: Néstor, Fernando, Inés y Luisa
- Nuestros colaboradores:
Romina, Pablo, Mauricio y Javier
- Coordinador
Emiliano Churruca
- Colaboración en materiales
Fabiana Conde
Silvia Silvetti



Objetivos de la materia

- Que se introduzcan como estudiantes universitarios (Aprender sus derechos y obligaciones a lo largo de la cursada)
- Que conozcan lo que son las ciencias de la computación y la informática y en qué parte cae la carrera que eligieron para cursar (formación, perfil de graduado, expectativas laborales a futuro, etc.)
- Que aprendan los elementos fundamentales de los problemas computacionales y su solución mediante programación (qué es un programa y qué problemas resuelve)
- Que conozcan las distintas estructuras que conforman un programa (estructuras de organización y control de flujo simple, en términos de comandos, y expresiones simples de tipo booleano)
- Entender las aplicaciones de la lógica proposicional y de predicados simples en términos de problemas computacionales (aplicando conectivas lógicas a condicionales de distinto tipo)

¿Qué esperamos de su parte?

- Responsabilidad:
 - Venir a todas las clases (Salvo causas de fuerza mayor).
 - Realizar todas las actividades que les planteamos.
 - Participar activamente de los espacios de trabajo en grupo y de resolución conjunta.
 - Entender que el objetivo es que aprendan, no solo aprobar (lo segundo es consecuencia de lo primero) y evitar intentar “tomar atajos”.
- Consultar todo:
 - Estamos todos para el mismo objetivo, que ustedes se lleven los conocimientos. Cada uno desde su rol.
 - Toda pregunta, por tonta que parezca, hay que hacerla. Estamos para responderlas.
 - Participá en las resoluciones conjuntas y compartí tus dudas y certezas.
 - Intervení cuando crees que hay una mejor solución, o cuando le pifiamos a algo (nos puede pasar)
- Horarios y dedicación:
 - Toda materia tiene una cantidad de horas de cursada y una cantidad de tiempo que hay que dedicarle en casa. En general hay que dedicarle más o menos el mismo tiempo en casa que en clase.
 - Esta materia son entonces por semana: 4 horas de clase (2 presenciales y 2 virtuales) y 4 horas que esperamos le dediquen en casa.



Fechas Importantes

PARCIAL 1:

- Semana 11-14 de septiembre

PARCIAL 2:

- Viernes 20 de octubre

RECU PARCIAL 1:

- Semana 25-28 de septiembre

RECU PARCIAL 2:

- Viernes 10 de noviembre

Integrador:

- Viernes 15 de diciembre



Criterios para regularizar la materia

- Se debe aprobar ambos exámenes parciales (en su primer instancia o su correspondiente recuperatorio)
- Cada examen se aprueba con nota igual o mayor a 4 (cuatro)
- Si se desaprueba (nota menor a 4) tanto un parcial como su recuperatorio, se pierde la cursada y deberá volver a cursar la materia.

Importante: Regularizar no es lo mismo que aprobar. Regularizar significa que no se debe volver a cursar la materia (salvo que no se apruebe, como veremos más adelante), pero no que esté aprobada.



Sobre exámenes

- Un 4 equivale aproximadamente al 60% del examen correcto. Es decir, es un poco más de la mitad del examen bien hecho.
- Sí se falta a la instancia parcial, se **debe** presentar al recuperatorio. No se anula la posibilidad de recuperar por tener ausente en la instancia parcial.
- Sí se desaprobó la instancia parcial, se **debe** presentar al recuperatorio.
- Se **puede** presentar a recuperar para levantar nota en los exámenes, pero la nota que queda es la del recuperatorio. Es un arma de doble filo, ya que si te va peor, la nota que queda es la del recuperatorio, incluso si se desaprueba, lo que implica perder la materia.
- Es importante prepararse para la instancia de parcial y no para el recuperatorio, ya que este tiende a ser un poco más difícil que el parcial.

Criterios de aprobación - parte 1

- Se debe tener ambos exámenes aprobados con nota igual o mayor a 6 (seis) y además que el promedio entre ambos sea igual o mayor a 7 (siete). Es decir, se acredita la materia y además las notas cumplen el criterio mencionado, lo que se conoce como **aprobar por promoción**.
 - Ejemplos de aprobación **por promoción**:
 - Examen 1: 7, Examen 2: 7
 - Examen 1: 6, Examen 2: 8
 - Examen 1: 8, Examen 2: 6
 - Ejemplos de regularización pero no de aprobación **por promoción**:
 - Examen 1: 5, Examen 2: 10
 - Examen 1: 9, Examen 2: 5

Importante: Cuando decimos Examen 1, o Examen 2 en los ejemplos, no significa necesariamente en la primera instancia de parcial, sino que puede ser también en el recuperatorio. Es decir, la nota que haya resultado de esa instancia de evaluación.

Criterios de aprobación - parte 2

- Sí se regulariza, pero no se aprueba **por promoción** (según los criterios mencionados en la slide anterior) se deberá rendir un examen adicional que integra todos los temas.
 - Examen integrador. Justo después de terminar la cursada. Se aprueba con 4 o más.
 - Examen final: Sí no se rindió integrador, o se rindió pero se desaprobó, se debe rendir un examen final. A esta instancia hay que anotarse (en las fechas de llamado de final). Se aprueba con nota 4 o más.

Importante: El integrador y el final son exámenes similares, en cuanto a temas, pero bastante más complejos y con mayor exigencia al momento de corrección. Precisamente se busca que demuestren que saben los temas de manera equivalente a quién aprobó por promoción.

Aclaraciones:

- Si se aprueba el integrador nota final que quedará para la materia resulta de promediar la obtenida en el integrador y ambos parciales.
- Una vez regularizada la materia se cuenta con el integrador y 10 finales seguidos (2 años) para aprobar. En caso de no lograrlo, se vence la regularidad de la materia, con lo cual se debe volver a cursar.



Sobre los exámenes de la materia

Por las características de la materia (y de las carreras en general) los temas son acumulativos. Es decir, que en el segundo examen se agregan temas, pero todo lo aplicado en el primer examen también aplica y se evalúa. Así, es imposible resolver los temas del segundo examen sin saber los del primero.

¡Por eso es importante llevar la materia al día!

De esta forma un examen integrador se parece un poco al segundo parcial, pues los temas se van acumulando, aunque tiene mayor complejidad.

¿Qué vine a aprender?

**Tiene que ver con
computadoras...**



Computadora

Una computadora es una máquina electrónica o electromecánica que recibe datos, los analiza, los procesa y los transforma, convirtiéndolos en información conveniente y útil para el posterior uso, en general por seres humanos.



Computadora

O sea, una computadora es una máquina capaz de transformar datos en información.

La gracia es que pueden hacer esto rápido, muy rápido. De hecho, la palabra computadora viene de computar, que significa calcular.

Las computadoras, en un nivel más elemental, lo único que hacen son cálculos matemáticos, muchos, muchísimos, muy rápidamente (del orden del billón por segundo).

Los datos de entrada pueden provenir de distintas fuentes (teclado y mouse, sensores, etc.) y los datos de salida pueden también presentarse en diversas formas (pantalla, impresora, parlantes, etc.). En la vida moderna, las computadoras están tan presentes, y hacen tantas cosas, que muchas personas desconocen su verdadera naturaleza.

Esta es una computadora que responde a nuestra definición



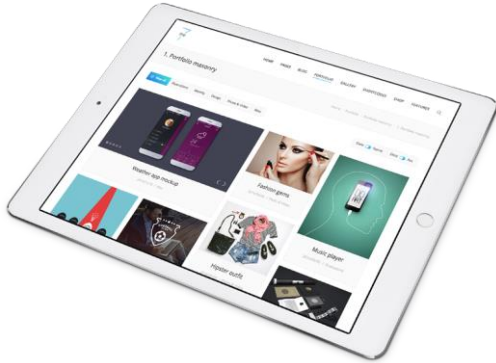
Pero también estas...





¿Se te ocurren otras?

¿Se te ocurren otras?







Hardware y Software

Claramente hay de muchas formas, y son muy distintas entre sí, en su función, en la forma de operarlas, etc.

Pero todas tienen partes internas similares. En particular hay dos aspectos claves de una computadora:

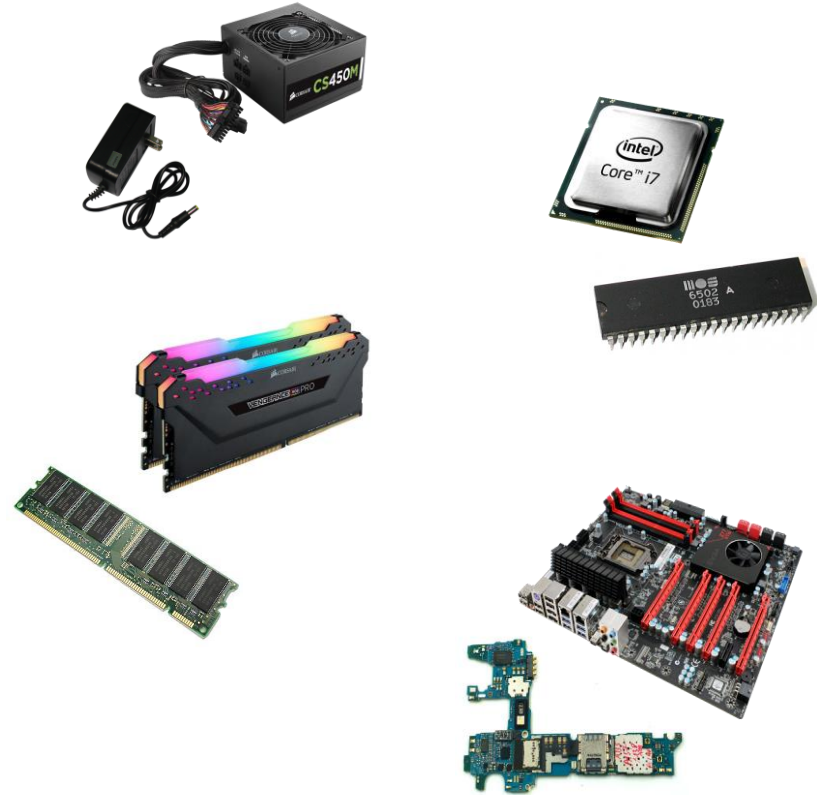
- **El Hardware:** El hardware comprende todos los elementos físicos que componen a la computadora. Es decir, el conjunto de circuitos, cables, perillas, palancas, botones, luces, displays, dispositivos de impresión, motores, imanes, placas metálicas, etc.
- **El Software:** El software es el conjunto de los programas de cómputo, archivos, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.

Como regla del pulgar, si no anda, y lo puedo patear, es hardware. Si no anda, y solo lo puedo insultar, es software.

Hardware

La mayoría de las computadoras poseen una serie de partes más o menos estándar en términos físicos:

- Fuente de alimentación
- CPU
- Memoria RAM
- Motherboard
- Periféricos

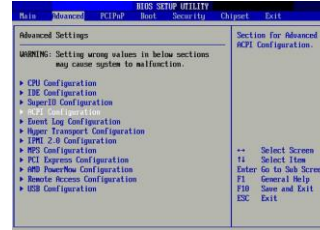


Software

Una computadora sin software no sirve para nada, pues sin electricidad, simplemente no funciona. Más aún, no solo debe haber electricidad, sino que debe haber electricidad en los lugares correctos en el momento correcto para que el equipo funcione de la forma esperada.

Entre el software al que podemos nombrar podemos mencionar:

- Firmware
- Sistema Operativo
- Programas
- Archivos



Sobre “programas”



¿Qué es un programa?

En general cuando hablamos de programa estamos hablando de un producto de software que es capaz de ejecutar en una computadora, y que resuelve un problema específico.

Es importante notar que “un programa” no solo está compuesto de ese software. Hay muchos otros productos, como archivos de configuración, documentación, manuales, etc. que también forman parte de “un programa”, pero nos vamos a centrar en lo que llamamos generalmente “aplicación” (ese ícono al que le damos doble clic y abre una pantalla en nuestra computadora para que hagamos algo específico).



¿Qué es un programa?

Un programa es un texto que describe la solución a un problema computacional y que puede ser ejecutado por una computadora.



¿Qué es un programa?

Un programa es **un texto** que describe la solución a un problema computacional y que puede ser ejecutado por una computadora.

En las computadoras modernas, un programa se escribe, en general en un archivo de texto. El texto describe (de alguna forma) cómo se soluciona un problema.



¿Qué es un programa?

Un programa es un texto que describe la solución a un **problema computacional** y que puede ser ejecutado por una computadora.

Un problema computacional es aquel que puede ser resuelto por una computadora. Por ej. sí mi problema es “mi padre no me quiere y nada de lo que haga satisface sus expectativas”, no voy a poder construir un programa que lo solucione, necesito probablemente años de terapia.

Un problema computacional entonces tiene que estar expresado como una transformación de información (recordemos, las computadoras toman información como entrada y producen información como salida).



¿Qué es un programa?

Un programa es un texto que describe la solución a un **problema computacional** y que puede ser ejecutado por una computadora.

“Dados 3 notas de un estudiante, calcular su promedio” es un problema computacional, hay información de entrada (las notas del estudiante) y se desea obtener una salida (el promedio). El programa describe entonces cómo se transforma esa entrada en la salida deseada.

Otros ejemplos podrían ser: “buscar la ruta con menos tránsito entre 2 puntos de un mapa”, “calcular cuál es la mejor inversión en bolsa según las últimas cotizaciones”, etc.



¿Qué es un programa?

Un programa es un texto que describe la solución a un problema computacional y que puede ser **ejecutado por una computadora**.

Para que un programa sea un programa, tiene que poder ser ejecutado por una computadora. Una descripción en español de como solucionar el problema, no alcanza, si una computadora no puede ejecutar dicha descripción.

Notar que el programa no soluciona el problema, la máquina, al ejecutar el programa descrito por el texto, es la que lo soluciona.

Intérpretes, Compiladores y Lenguajes de Programación

¿Cómo se llega de un texto a que la computadora lo ejecute?

¿Qué distingue a un archivo de texto con código de un programa ejecutable?

Hay tres formas distintas de que ese programa se vuelva ejecutable.

- Mediante un intérprete
- Mediante un compilador
- Mediante alguna combinación de las anteriores.



Intérprete

Un intérprete es una aplicación que es capaz de leer el texto de un programa y luego realizar en la computadora cálculos asociados.

Por ejemplo, si nuestro programa dice algo como “sumar 5 y 7”, el intérprete es capaz de leer ese texto y realizar “5+7” en el CPU de la computadora.

El intérprete no entiende cualquier cosa. Para una computadora no es lo mismo decir

- “suma 5 y 7”
- “sumar 5, 7”
- y “sumar 5 y 7”.

Aunque las personas lo entendamos como sinónimos de la misma idea.



Lenguajes

Un lenguaje no es más que un conjunto de símbolos y reglas que usamos para comunicarnos.

El español, el inglés, el catalán, son lenguajes. Los símbolos que usamos (las letras y signos de puntuación) no son idénticos en cada uno de ellos (la Ñ, por ejemplo, es única del español). Las reglas de composición (palabras, reglas gramaticales) tampoco son iguales.

El problema con esos lenguajes es que son ambiguos. Hay muchas formas de decir lo mismo, y hay cosas que decimos, que solo pueden ser entendidas mediante un análisis del contexto.

Cómo se interpreta la frase “lo voy a tomar”



Lenguajes

Cómo se interpreta la frase “lo voy a tomar”.

- Puedo estar hablando de un vaso de un líquido
- Puedo estar hablando de un medio de transporte
- Puedo estar hablando de “asir” o “agarrar” algo.

La frase sólo se entiende en el contexto.

Las computadoras no entienden de contexto, necesitan lenguajes no ambiguos.



Lenguajes de programación

Un lenguaje de programación es un lenguaje, que no es ambiguo, y cuya función es permitir al programador comunicarse con la computadora.

Cuando escribimos un programa lo hacemos en un lenguaje de programación específico, y ese lenguaje asociará una sintaxis (símbolos y reglas) con una semántica (una interpretación sobre qué es lo que debe entenderse al leer una secuencia de símbolos determinada).

El lenguaje es lo que nos va a decir cómo debe escribirse la semántica “ $5+7$ ”. Según las reglas que disponga podrá ser “Sumar 5 y 7”, “sumar 5, 7”, “sum 5 7” o cualquier cosa que se nos ocurra.



Intérpretes (cont.)

- Un intérprete está pensado para comprender un lenguaje de programación específico. O sea, habrá un intérprete para Python, un intérprete para PHP, etc.
- El texto del programa a interpretar tiene que cumplir con las reglas planteadas por ese lenguaje de programación, para que el programa pueda ser interpretado de forma adecuada.
- Los creadores de un lenguaje de programación suelen proveer herramientas específicas para trabajar con el lenguaje, lo cual puede incluir un intérprete.



Compiladores

El intérprete no es la única forma de tener aplicaciones ejecutables a partir de un programa. Otra forma es la compilación.

Los archivos poseen lo que se conoce como “cabecera” que le indica al sistema operativo cómo debe interpretarlo. En algunos sistemas operativos está dado por la **extensión de archivo**, un sufijo (en general oculto a simple vista) del nombre de un archivo en la computadora.

Por ej. **mi archivo.txt** es un archivo que contiene texto (.txt es la extensión para textos) mientras que **programa.exe** es un archivo de un programa ejecutable (.exe es la extensión para programas)

Por supuesto, el binario asociado a un programa como texto, y a un programa como un programa ejecutable suele ser muy distinto (usan una codificación distinta del binario al momento de interpretarse).



Compiladores

Un compilador es un programa capaz de tomar un archivo de texto que contiene un programa escrito en un lenguaje de programación específico, y producir un archivo de aplicación ejecutable.

Decimos que el texto que describe el programa es el código fuente de un programa. Una versión de aplicación ejecutable generada a partir de esa descripción es el código objeto o código ejecutable del programa.

Un compilador transforma de código fuente a código objeto.

Nuevamente, algunos lenguajes incluyen entre su conjunto de herramientas un compilador.



¿Lenguajes compilados o interpretados?

Un lenguaje de programación no es ni interpretado, ni compilado. El mismo programa puede interpretarse o compilarse, según la herramienta que se use.

En general mucha gente habla erróneamente de “lenguajes compilados” o “lenguajes interpretados” porque los creadores del lenguaje suelen incluir una de las dos herramientas (y no ambas). Pero nada impide que se genere un intérprete para un lenguaje que se provee con un compilador, o viceversa.

Más aún, muchas herramientas hacen cosas intermedias. Por ejemplo, realizan una pre-compilación a un lenguaje intermedio, más simple de interpretar, y luego interpretan ese código intermedio (Python y Java son casos de esto).

Paradigmas de programación



Paradigmas de programación

Los lenguajes de programación son los que determinan cómo se escribe un programa en código.

El código, sin embargo, intenta reflejar abstracciones de ideas que están en la mente del programador. Y cómo tales, estas ideas pueden ser muy dispares pues provienen de lugares muy distintos.

Un paradigma de programación es una forma de concebir un programa (en términos abstractos) para describir la solución a un problema computacional.

Existen muchos paradigmas, algunos más estandarizados y difundidos que otros.



Paradigma de Programación Estructurada

Los programas de este paradigma tienen como concepción fundamental la idea de que hay dos elementos clave:

- Los comandos (que describen acciones)
- Las expresiones (que describen datos)

Un programa es una combinación de comandos y expresiones, que se disponen con una estructura secuencial. Además, existen elementos que permiten cambiar el flujo secuencial de un programa, para obtener flujos distintos.

A veces también se le llama paradigma imperativo (aunque hay una diferencia entre la concepción imperativa y la estructurada) La parte clave de este último se basa en plantear estructuras (subtareas) que permiten comunicar claramente las ideas del problema a resolver, y combinarlas.



Paradigma de Programación Orientada a Objetos

Este paradigma extiende las ideas del paradigma estructurado, y plantea entidades abstractas llamadas objetos, que se comunican entre sí, enviando mensajes, para arribar a la solución a un problema.

Las ideas de comando y expresiones siguen presentes, en la forma de “mensajes que piden hacer cosas” y “mensajes que piden responder cosas”.

Otras abstracciones adicionales se incorporan en este paradigma para lograr programas sumamente complejos con una estructura práctica de entender.



Paradigma Funcional

Este paradigma plantea un modelo más matemático. Los comandos no existen, sino que todo son expresiones.

La unidad elemental es la función, que plantea una equivalencia entre una definición y otra.

Ejecutar un programa consiste entonces en reducir una expresión en otras, hasta arribar a valores concretos.



Paradigma Lógico

Este paradigma plantea un modelo lógico, en dónde el programador plantea axiomas, y luego el programa es una consulta.

Sí la consulta puede responderse a partir de los axiomas, entonces se obtiene una respuesta.

Es un paradigma muy distinto a los anteriores, y que es ideal para algunos problemas puntuales.



Otros paradigmas

Existen muchos otros paradigmas, menos extendidos y difundidos, como el paradigma orientado a aspectos, o el orientado a eventos.

Algunos de estos son muy recientes, y no han tenido aún tracción. Otros, están en debate sobre si realmente se trata de nuevos paradigmas, o son solo agregados mínimos a paradigmas existentes, que no implican una concepción distinta sobre lo que significa un programa.


De cualquier forma, lo que sí es cierto es que los paradigmas no surgieron de un repollo. El paradigma imperativo está basado en las ideas de Alan Turing, y está más pegado a los circuitos de la máquina. El funcional se basa en los estudios de Alonzo Church ya en los años 1930s, y derivó en el lógico ya en los 60s y 70s. La programación orientada a objetos surge como una alternativa ya en los 1980s, y se populariza en los 90s.

En definitiva, nada implica que no puedan aparecer nuevas ideas y paradigmas, o que estos se vayan especializando y mejorando.


**En esta materia vamos a
aprender algunos conceptos
fundamentales de la
programación**

**Vamos a trabajar con bloques
encastrables, y luego en texto,
y el objetivo es aprender a
interpretar y enfrentar
problemas computacionales**

**La lógica va a estar presente
en todos los programas que
realicemos.**



Momento de dudas o consultas



Qué es programar y diferentes formas de programación

Nos vemos la próxima