



Ministerio de Cultura y Educación
 Universidad Nacional de San Luis
 Facultad de Ciencias Físico Matemáticas y Naturales
 Departamento: Informatica
 Area: Departamental

(Programa del año 2019)

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
INTRODUCCION A LA COMPUTACION	LIC.CS.COMP.	32/12	2019	1° cuatrimestre
INTRODUCCION A LA COMPUTACION	TFA	04/14	2019	1° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
DORZAN, MARIA GISELA	Prof. Responsable	P.Adj Exc	40 Hs
RESCALDANI, LUCRECIA MONICA	Responsable de Práctico	JTP Semi	20 Hs
CAVECEDO ALMANZA, GABRIEL ALEJ	Auxiliar de Práctico	A.2da Simp	10 Hs
TRUGLIO, MATIAS IVAN	Auxiliar de Práctico	A.1ra Semi	20 Hs

III - Características del Curso

Credito Horario Semanal				
Teórico/Práctico	Teóricas	Prácticas de Aula	Práct. de lab/ camp/ Resid/ PIP, etc.	Total
0 Hs	2 Hs	3 Hs	2 Hs	7 Hs

Tipificación	Periodo
B - Teoria con prácticas de aula y laboratorio	1° Cuatrimestre

Duración			
Desde	Hasta	Cantidad de Semanas	Cantidad de Horas
13/03/2019	21/06/2019	15	105

IV - Fundamentación

La mayoría de los estudiantes que ingresan a primer año de las carreras relacionadas con las Ciencias de la Computación lo hacen sin saber programar, pese a ser nativos digitales y que las computadoras (y demás dispositivos) poseen muchos programas y aplicaciones que forman parte de su vida cotidiana.

Por ser esta asignatura, la primera específica de programación, se plantean entonces varios desafíos: enseñar una metodología para la resolución de problemas e introducir un lenguaje formal simple para escribir los programas focalizando la enseñanza de los conceptos algorítmicos. Además, se debe fomentar un ambiente de trabajo que despierte el espíritu emprendedor de los estudiantes evitando que se sientan abrumados o desanimados en este primer encuentro con la programación. Alcanzando el éxito en estos desafíos se trata de evitar la deserción temprana de los estudiantes.

V - Objetivos / Resultados de Aprendizaje

El objetivo general es resolver problemas básicos a través de la construcción de programas basados en algoritmos, es decir, definir un conjunto de pasos lógicamente ordenados, precisos y no ambiguos, escritos a través de símbolos o en lenguaje natural siguiendo una metodología de trabajo adecuada. Si bien el conjunto de símbolos, instrucciones y estructuras presentes en un algoritmo o programa son fáciles de identificar y aprender en relación a su significado, la dificultad se presenta al intentar combinar lógicamente estas instrucciones y estructuras para que resuelvan un problema planteado. Por lo tanto, se plantean los siguientes objetivos específicos:

- Desarrollar una adecuada metodología de trabajo para la resolución de los problemas introduciendo estrategias para resolver

problemas, como por ejemplo, dividir un problema en subproblemas, obtener la solución a través de refinamientos sucesivos, entre otras.

- Generar la capacidad necesaria para saber interpretar claramente los objetivos del problema y poder resolverlo, es decir, identificar las posibles restricciones o condiciones que deben ser consideradas en la resolución del problema.
- Desarrollar la capacidad de diseñar un algoritmo que modele la resolución del problema, de implementar el algoritmo en un lenguaje de programación y de elegir un conjunto adecuado y representativo de valores de los datos de entrada para realizar una correcta prueba del algoritmo realizado.
- Introducir la notación lógica formal con el fin de expresar ideas o razonamientos de forma clara y precisa, fomentando la rigurosidad y formalidad.
- Introducir conceptos formales de especificación y verificación de algoritmos, describiendo la transformación de los datos del entorno a través de pre y postcondiciones.
- Reforzar y desarrollar competencias generales, como por ejemplo: trabajo en equipo, comunicación en forma oral y escrita, aprendizaje autónomo, entre otras.
- Promover el uso de buenos hábitos de programación incentivando, desde el principio de la carrera, el ejercicio de la documentación, los comentarios y la indentación de los programas desarrollados.

VI - Contenidos

UNIDAD 1: LÓGICA PROPOSICIONAL

El lenguaje de las proposiciones: Alfabeto y sintaxis de las fórmulas bien formadas. Construcción de enunciados. Conjunto de significados. Clasificación de fórmulas bien formadas. Equivalencia lógica.

UNIDAD 2: RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

Comprensión de problemas en general. Método de 4 etapas de Polya basado en preguntas. Representación de problemas: abstracciones. Resolución de problemas y computadoras. Metodología a desarrollar para la resolución de un problema por medio de algoritmos: formulación del problema, diseño de algoritmos, codificación y ejecución. Desagregación del problema. Metodología de refinamiento por pasos sucesivos.

UNIDAD 3: LENGUAJE DE DISEÑO DE ALGORITMOS - TIPOS DE DATOS, OPERACIONES Y EXPRESIONES - INTERACCIÓN: ENTRADA Y SALIDA DE DATOS

Formalización concepto de algoritmo. Ambiente de un algoritmo. Datos de entrada y salida. Transformación del ambiente. Objetos del ambiente: constantes y variables. Tipos de datos primitivos. Expresiones. Operadores aritméticos, lógicos y relacionales. Precedencia y orden de evaluación. Funciones primitivas. Operación de asignación. Estructura general de un programa. Definición de variables. Operaciones de entrada y salida de datos. Cuestiones de estilo: indentación, elección de nombres adecuados de variables, comentarios.

UNIDAD 4: LENGUAJE DE DISEÑO DE ALGORITMOS - ESTRUCTURAS DE CONTROL

Estructura de control secuencial: Concepto. Estructura de control condicional: Simple y Múltiple. Elección entre dos alternativas. Elección entre varias alternativas. Estructuras de control repetitiva: Conceptos. Estructuras con número de iteraciones predeterminado y no predeterminado. Pautas para seleccionar la estructura repetitiva más adecuada. Ciclos infinitos. Anidamiento de estructuras de control. Ejecución de un programa a través de tablas de ejecución. Utilización de diagramas de flujo para visualizar los posibles flujos de ejecución de un algoritmo.

UNIDAD 5: LENGUAJE DE DISEÑO DE ALGORITMOS - ESTRUCTURAS DE DATOS

Definición de estructura de datos. Tipo de datos estructurados versus tipos de datos simples. Arreglos lineales. Índice y componentes. Operaciones sobre arreglos lineales: asignación, recuperación y recorrido.

UNIDAD 6: LENGUAJE DE DISEÑO DE ALGORITMOS - SUBALGORITMOS

Definición de subalgoritmos. Ambiente de un subalgoritmo. Funciones y procedimientos. Parámetros actuales y formales. Tipo de pasaje de parámetros. Invocación de subalgoritmos.

UNIDAD 7: LÓGICA DE PREDICADOS

Limitaciones de la Lógica Proposicional. Lenguaje de Predicados: alfabeto, términos y predicados. Cuantificador existencial

y universal. Uso de los cuantificadores. Interpretación en lógica de predicados.

UNIDAD 8: INTRODUCCIÓN A LA ESPECIFICACIÓN Y VERIFICACIÓN FORMAL DE ALGORITMOS

Especificación de algoritmos. Pre/Post condiciones. Verificación de Algoritmos. Reglas generales de verificación. Precondición más débil. Algunas reglas específicas.

VII - Plan de Trabajos Prácticos

Los trabajos prácticos de aula/laboratorio consisten en la resolución de problemas por parte de los estudiantes y asistidos por los docentes, correspondientes a las unidades temáticas del programa. Las actividades evaluativas solicitadas en la asignatura son pensadas con la finalidad de ir marcando “instantáneas en el tiempo”, tanto en el proceso de aprendizaje, como en el de enseñanza para que le den al estudiante y al docente información acerca de la situación particular antes de llegar a la instancia escrita o integradora.

Al comenzar cada trabajo práctico los estudiantes deben contestar una pequeña evaluación con ítems múltiple opción y preguntas de respuestas simples relacionados con la teoría correspondiente a dicho trabajo práctico. Se pretende que el estudiante comience cada trabajo práctico habiendo leído los apuntes teóricos. Además, con este tipo de actividad, se busca que el estudiante empiece a expresarse de manera escrita, a socializar y reflexionar sobre sus respuestas y a comprender cuáles fueron las debilidades marcadas en la corrección informada.

En general se proponen problemas que generen dudas y motiven la consulta a los docentes. La idea general es que los estudiantes resuelvan los ejercicios y utilicen el horario de práctica para realizar consultas surgidas sobre los mismos. Se puede utilizar la herramienta PsFlex la cual fue pensada y diseñada por el equipo de cátedra y desarrollada por Gabriel Cavecedo, también integrante del equipo de cátedra. Dicha herramienta sirve para asistir al estudiante en sus primeros pasos en programación y realizar ejercicios directamente en la computadora o corroborar que los ejercicios realizados en el papel son correctos. Se diseñó utilizando el Lenguaje de Diseño de Algoritmo, el cual posee una sintaxis simple e intuitiva permite centrar la atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas.

Para realizar un seguimiento en el nivel de apropiación de los conocimientos de los estudiantes, se requiere la participación activa de ellos en las clases.

En todas estas actividades evaluativas, no se exige la aprobación, sino que se pretende que el estudiante se enfrente con ejercicios para que pueda medir su nivel apropiación del conocimiento, realizando en cada una de ellas una corrección informada del docente hacia el estudiante ya que lo ayuda razonablemente a aprender. En este sentido es que también se brindan consultas en la oficina tratando de que pierdan el miedo a preguntar. Quizás en un ambiente diferente al del aula, pierdan ese miedo o timidez a preguntar.

Para integrar los conceptos adquiridos durante la cursada de la asignatura, se solicita el desarrollo de un proyecto final integrador programado en PsFlex, el cual debe realizarse en grupo para poder compartir experiencias.

Las vías de comunicación con los estudiantes son las siguientes:

- Correos electrónicos: mgdorzan@unsl.edu.ar, rescalda@unsl.edu.ar, mitrugli@unsl.edu.ar, gcavecedo@unsl.edu.ar
- Oficina: 30 – 1° piso – 2° Bloque
- Sitio web: <http://www.dirinfo.unsl.edu.ar/intcomp/>
- Campus Virtual: <http://www.evirtual.unsl.edu.ar/moodle/> donde se pueden descargar las teorías, trabajos prácticos, avisos importantes, apuntes, cronograma, hacer consultas o comentarios, etc.
- Facebook: <https://www.facebook.com/Introducción-a-la-Computación-UNSL-446411485435846/> donde se colocan avisos importantes y se pueden realizar consultas o enviar mensajes.

TRABAJO PRÁCTICO 1: LÓGICA PROPOSICIONAL

Objetivos generales: Introducir la notación lógica formal con el fin de expresar ideas o razonamientos de forma clara y precisa, fomentando la rigurosidad y formalidad.

Objetivos específicos: Identificar el alfabeto y la sintaxis de las fórmulas bien formadas. Poder construir fórmulas bien formadas a partir de enunciados expresados en lenguaje coloquial. Determinar el valor de verdad de fórmulas bien formadas utilizando las tablas de verdad, es decir, el conjunto de significados de las fórmulas bien formadas. Determinar la equivalencia lógica de fórmulas bien formadas utilizando las tablas de verdad.

TRABAJO PRÁCTICO 2: RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

Objetivos generales: Interpretar claramente los objetivos del problema y poder resolverlo, es decir, identificar las posibles restricciones o condiciones que deben ser consideradas en la resolución del problema. Aplicar una adecuada metodología de trabajo para la resolución de los problemas introduciendo diferentes estrategias para resolver problemas.

Objetivos específicos: Crear modelos y abstracciones de problemas de la vida cotidiana. Dado un problema identificar los datos de entrada y los datos de salida como así también las restricciones del mismo. Descomponer problemas complejos en tareas más sencillas con el objeto de resolver el problema original. Diseñar algoritmos que resuelvan problemas utilizando un conjunto finito de instrucciones y la posterior ejecución de dichas instrucciones. Con la idea de motivarlos, fomentar la curiosidad y su capacidad de indagación, los estudiantes realizan ejercicios los cuales pueden ser probados en el sitio web de PilasBloques (<http://pilasbloques.program.ar/online/#/desafios>) la cual es una herramienta desarrollada por Program.AR y y en el sitio web de Gobstones (<https://gobstones.github.io/gobstones-sr/>) el cual es un lenguaje desarrollado específicamente para la enseñanza de un curso introductorio a la programación.

TRABAJO PRÁCTICO 3: LENGUAJE DE DISEÑO DE ALGORITMOS – TIPOS DE DATOS, OPERACIONES Y EXPRESIONES - INTERACCIÓN: ENTRADA Y SALIDA DE DATOS

Objetivos generales: Escribir expresiones del lenguaje de diseño, analizando sintaxis y significado. Traducir enunciados en lenguaje natural a expresiones aritméticas, lógicas y relacionales. Comprender la interacción con usuario.

Objetivos específicos: Describir el ambiente de un algoritmo: variables de entrada, salida y auxiliares. Identificar los objetos variables y constantes de un problema. Declaración de variables en el algoritmo.

Identificar los tipos de datos primitivos que provee el lenguaje de diseño y las operaciones definidas para cada tipo de dato. Construir expresiones en lenguaje de diseño. Evaluar diferentes tipos de expresiones. Identificar el mecanismo de trabajo del operador de asignación. Asignaciones válidas e inválidas. Desarrollar algoritmos que involucran la definición de variables y la lectura y escritura de datos. Ejecución de algoritmos utilizando tablas de ejecución.

TRABAJO PRÁCTICO 4: LENGUAJE DE DISEÑO DE ALGORITMOS – ESTRUCTURAS DE CONTROL CONDICIONAL

Objetivos generales: Comprender el funcionamiento de la estructura de control condicional simple y múltiple, introduciendo la importancia de realizar pruebas para comprobar el funcionamiento de un algoritmo.

Objetivos específicos: Desarrollar algoritmos que involucran el uso de las estructuras de control condicional: simple y múltiple. Identificar cuándo es adecuado la utilización de cada estructura. Ejecutar algoritmos con diferentes estructuras de control condicional, haciendo hincapié en la selección de un adecuado conjunto de casos de prueba. Realización de diagramas de flujo para visualizar los diferentes caminos de ejecución de un algoritmo.

TRABAJO PRÁCTICO 5: LENGUAJE DE DISEÑO DE ALGORITMOS - ESTRUCTURAS DE CONTROL REPETITIVO

Objetivos generales: Comprender el funcionamiento de estructuras de control repetitivo.

Objetivos específicos: Comprender el mecanismo de trabajo de una estructura de control repetitivo. Identificar los casos especiales en cada estructura. Identificar cuándo es adecuado la utilización de cada estructura. Ejecutar algoritmos con diferentes estructuras de control repetitivo. Desarrollar algoritmos que involucran el uso de tres estructuras de control repetitivo. Realización de diagramas de flujo para visualizar los diferentes caminos de ejecución de un algoritmo.

TRABAJO PRÁCTICO 6: LENGUAJE DE DISEÑO DE ALGORITMOS - ESTRUCTURAS DE DATOS

Objetivos generales: Comprender y aplicar los conceptos relacionados con la estructura de datos.

Objetivos específicos: Identificar la necesidad de utilizar uno o más arreglos lineales para resolver un problema. Declarar un arreglo en lenguaje de diseño. Identificar los límites de un arreglo lineal. Conocer las operaciones válidas sobre las componentes de un arreglo: asignación de un valor y consulta del valor de un elemento del arreglo.

TRABAJO PRÁCTICO 7: LENGUAJE DE DISEÑO DE ALGORITMOS - SUBALGORITMOS

Objetivos generales: Comprender y aplicar los conceptos relacionados con la modularización de procesos.

Objetivos específicos: Definir subalgoritmos. Identificar el ambiente de un subalgoritmo. Identificar los tipos de parámetros y el pasaje de parámetros. Definir funciones y procedimientos. Identificar cuando es adecuado el uso de cada tipo de subalgoritmo. Invocar subalgoritmos.

TRABAJO PRÁCTICO 8: LÓGICA DE PREDICADOS - INTRODUCCIÓN A LA ESPECIFICACIÓN Y VERIFICACIÓN FORMAL DE ALGORITMOS

Objetivos generales: Comprender y aplicar los conceptos relacionados con la Lógica de Predicados. Comprender y aplicar los conceptos relacionados con la especificación y verificación formal de algoritmos.

Objetivos específicos: Identificar las limitaciones de la Lógica Proposicional. Identificar el alfabeto, términos y predicados de la lógica de predicados. Utilizar cuantificadores. Construir especificaciones formales de algoritmos utilizando pre/post condiciones. Leer especificaciones formales de algoritmos para saber qué realiza dicho algoritmo. Aplicar reglas de verificación formal generales y específicas.

TRABAJO PRÁCTICO 9:

Realizar un proyecto final integrador utilizando PsFlex.

VIII - Regimen de Aprobación

1- Acerca de las condiciones de regularización de la asignatura.

Para regularizar la asignatura el estudiante debe cumplimentar los siguientes ítems:

a) Asistencia: concurrir al menos al 80% de las actividades previstas.

b) Actividades teórico-prácticas: entregar al menos el 80% en tiempo y forma.

Actividades teóricas: preguntas de respuesta corta y respuestas múltiple opción al inicio de cada trabajo práctico

Actividades prácticas: participación activa en clase y entrega de cierto ejercicios de cada trabajo práctico

c) Aprobar un proyecto final integrador (grupal)

d) Exámenes: aprobar el examen o alguna de sus dos respectivas recuperaciones con nota menor que 7 pero superior o igual a 6.

2- Acerca de la aprobación de la asignatura.

Existen dos formas de aprobación de la asignatura:

a) Por Promoción, para lo cual se exige la regularización de la asignatura aprobando el examen o alguna de sus dos respectivas recuperaciones con nota 7 o superior en cada uno, y la aprobación de un coloquio final integrador (oral o escrito) con nota mayor o igual a 7. La calificación final provendrá del promedio de las calificaciones obtenidas en el examen y/o recuperaciones aprobadas, en las actividades teóricas y prácticas, del Proyecto Final y del coloquio.

b) Por Regularización más examen final.

3- Acerca del examen final.

Dicho examen podrá ser oral o escrito y se rinde en turnos de exámenes establecidos en el Calendario Académico.

4- Acerca del examen libre.

Los estudiantes que no cumplen con los requisitos del régimen promocional o regular podrán rendir la asignatura como estudiantes libres. Para ello deberán entregar un práctico de aula, rendir un examen escrito sobre temas teóricos y prácticos, siendo obligatoria la aprobación de la parte práctica para considerar la parte teórica.

IX - Bibliografía Básica

[1] M. J. Abásolo, F. Guerrero y J. Perales López. "Introducción a la Programación", Colección materiales didácticos, Universidad de las Islas Baleares, 2011.

[2] E. P. Arís, J. L. Sánchez González y F. M. Rubio. "Lógica Computacional", Thomson Editores, España, 2003.

[3] G. Polya. "Cómo plantear y resolver problemas", Editorial Trillas, México, 1970.

[4] S. Braustein y A. Gioia. "Introducción a la Programación y a las Estructuras de Datos", Eudeba, Argentina, 1986.

[5] J. J. Garcia Molina, J. Fernandez Aleman, M. J. Majado Rosales y J. Montoya Dato Francisco. "Una Introducción a la Programación - Un Enfoque Algorítmico", Editorial: PARANINFO, 2005.

[6] F. Pinales Delgado y C. Velázquez Amador. "Problemario de algoritmos resueltos con diagramas de flujo y pseudocódigo". Universidad Autónoma de Aguascalientes. ISBN: 978-607-8285-96-9

[7] PsFlex (<http://psflex.forjota.com.ar/>)

[8] Material de estudio diseñado por el equipo docente.

X - Bibliografía Complementaria

[1] Z. Michalewicz y M. Michalewicz. "Puzzle Based Learning: Introduction to critical thinking, mathematics, and problem solving". Hybrid Publishers, 2008.

[2] "Introducción a la Informática". Universidad Nacional de Santiago del Estero Facultad de Ciencias Exactas y Tecnologías. 2011

[3] G. Sarria y Mario Mora, "Introducción a la Programación". 2012

[4] D. Gil, "Guía teórica. Informática Básica". 2007

XI - Resumen de Objetivos

- Desarrollar una adecuada metodología de trabajo para la resolución de los problemas introduciendo estrategias para resolver problemas.

- Desarrollar la capacidad de diseñar un algoritmo que modele la resolución del problema, de implementar el algoritmo en un lenguaje de programación y de elegir un conjunto adecuado y representativo de valores de los datos de entrada para realizar una correcta prueba del algoritmo realizado.

- Introducir la notación lógica formal con el fin de expresar ideas o razonamientos de forma clara y precisa, fomentando la rigurosidad y formalidad.

- Introducir conceptos formales de especificación y verificación de algoritmos, describiendo la transformación de los datos del entorno a través de pre y postcondiciones.

XII - Resumen del Programa

UNIDAD 1: LÓGICA PROPOSICIONAL

UNIDAD 2: RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

UNIDAD 3: LENGUAJE DE DISEÑO DE ALGORITMOS - TIPOS DE DATOS, OPERACIONES Y EXPRESIONES - INTERACCIÓN: ENTRADA Y SALIDA DE DATOS

UNIDAD 4: LENGUAJE DE DISEÑO DE ALGORITMOS - ESTRUCTURAS DE CONTROL

UNIDAD 5: LENGUAJE DE DISEÑO DE ALGORITMOS - ESTRUCTURAS DE DATOS

UNIDAD 6: LENGUAJE DE DISEÑO DE ALGORITMOS - SUBALGORITMOS

UNIDAD 7: LÓGICA DE PREDICADOS

UNIDAD 8: INTRODUCCIÓN A LA ESPECIFICACIÓN Y VERIFICACIÓN FORMAL DE ALGORITMOS

XIII - Imprevistos

XIV - Otros