|  |  |  |
| --- | --- | --- |
| **NOMBRE DE LA ENTIDAD:** |  | Campus Guanajuato, División de Ciencias Naturales y Exactas |

|  |  |  |
| --- | --- | --- |
| **NOMBRE DEL PROGRAMA EDUCATIVO:** |  | Licenciatura en Matemáticas |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NOMBRE DE LA UNIDAD DE APRENDIZAJE:** |  | Programación y Algoritmos I |  | **CLAVE:** |  | NELI06096 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FECHA DE APROBACIÓN:** |  |  |  | **FECHA DE ACTUALIZACIÓN:** |  |  |  | **ELABORÓ:**  |  | **Sergio Ivvan Valdez** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **HORAS DE TRABAJO****DEL ESTUDIANTE CON EL PROFR.:** |  | 72 |  | **HORAS DE TRABAJO AUTÓNOMO DEL ESTUDIANTE:** |  | 78 |  | **CRÉDITOS:** |  | 6 |
| **HORAS SEMANA/SEMESTRE** |  | **4** |  | **HORAS TOTALES DE TRABAJO DEL ESTUDIANTE:** |  | 150 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PRERREQUISITOS NORMATIVOS:** |  | Ninguno |  | **PRERREQUISITOS RECOMENDABLES:** |  | Ninguno |

|  |
| --- |
| **CARACTERIZACIÓN DE LA UNIDAD DE APRENDIZAJE** |
| **POR EL TIPO DE ACTIVIDADES DE APRENDIZAJE:** | **DISCIPLINARIA** | X | **FORMATIVA** |  | **METODOLÓGICA** |  |  |
| **POR SU UBICACIÓN EN LAS ÁREAS DE ORGANIZACIÓN CURRICULAR:** | ÁREA GENERAL |  | **ÁREA BÁSICA COMÚN** |  | **ÁREA DISCIPLINAR** | **X** | **ÁREA DE PROFUNDIZACIÓN** |  | **ÁREA COMPLEMENTARIA** |  |
| ÁREA NUCLEAR |  | **ÁREA DE INVESTIGACIÓN** |  | **ÁREA PROFESIONAL** |  |  |  |  |  |
| **POR LA MODALIDAD DE ABORDAR EL****CONOCIMIENTO:** | **CURSO** | X | **TALLER** |  | **LABORATORIO** |  | **SEMINARIO** |  |
| **POR EL CARÁCTER DE LA UDA:** | **OBLIGATORIA** |  | **RECURSA-BLE** |  | **OPTATIVA** | X | **SELECTIVA** |  | **ACREDITABLE** |  |

|  |
| --- |
| **PERFIL DEL DOCENTE:** |
| Para la impartición de esta unidad de aprendizaje se sugiere la participación de un doctor en Matemáticas, Ciencias de la Computación o áreas afines. |
| **CONTRIBUCIÓN DE LA UNIDAD DE APRENDIZAJE AL PERFIL DE EGRESO DEL PROGRAMA EDUCATIVO:** |
| La Unidad de Aprendizaje incide de manera directa en la formación de las competencias genéricas institucionales siguientes:CG1. Planifica su proyecto educativo y de vida de manera autónoma bajo los principios de libertad, respeto, responsabilidad social y justicia para contribuir como agente de cambio al desarrollo de su entorno.CG2. Se comunica de manera oral y escrita en español y en una lengua extranjera para ampliar sus redes académicas, sociales y profesionales que le permitan adquirir una perspectiva internacional.CG3. Maneja ética y responsablemente las tecnologías de la información para agilizar sus procesos académicos y profesionales de intercomunicación. Contribuye a las competencias específicas siguientes: CE2. Analiza, construye y desarrolla argumentaciones lógicas con una identificación clara de hipótesis y conclusiones para la resolución de problemas.CE3. Domina los conceptos elementales de la matemática clásica y su evolución histórica como parte fundamental de su desarrollo profesional. CE4. Conoce y aplica los conceptos elementales de la matemática moderna en diversas áreas del conocimiento CE6. Desarrolla disciplina de trabajo y capacidad de colaboración dentro de las matemáticas, así como con profesionales de otras áreas. CE7. Selecciona y conoce la herramienta matemática y/o computacional para resolver problemas en diferentes áreas del conocimiento.CE8. Explora temas avanzados de la matemática bajo la orientación de especialistas abriendo la opción de continuar con estudios de posgrado. |
| **CONTEXTUALIZACIÓN EN EL PLAN DE ESTUDIOS:** |
| La importancia de esta Unidad de Aprendizaje reside en que permite al estudiante profundizar en temas avanzados de Programación y Algoritmos para aplicarlos en la resolución de problemas de distintas áreas de las matemáticas. Esta Unidad de Aprendizaje forma parte del área disciplinar porque aporta elementos importantes para el ejercicio de la profesión.Al ser Unidades de Aprendizaje optativas, con ayuda del tutor, el alumno puede elegir el momento apropiado para cursarlas. Se relaciona con las materias del grupo de Computación. |
| **COMPETENCIA DE LA UNIDAD DE APRENDIZAJE:** |
| Explora temas avanzados de Computación bajo la orientación de especialistas, para profundizar sus conocimientos en el área. |

|  |
| --- |
| **CONTENIDOS DE LA UNIDAD DE APRENDIZAJE:** |
| 1. Introducción: Estructura y sintaxis de un programa en C; Archivos de cabecera; Palabras reservadas; Tipos de datos; Variables, declaración; Alcance de las variables, locales y globales; Tipos de expresiones. Lvalue, rvalue; Operadores aritméticos, lógicos, relacionales y misceláneos; printf y scanf; Compilación; Redirección de la entrada y salida estándar; Errores comunes; Tiempo de compilación y tiempo de ejecución; argv y argc.
2. Funciones: Declaración y definición de funciones; Partes de una función; Paso de argumentos por copia; Archivos cabecera propios; Compilación con varios archivos de declaración y definición de funciones.
3. Ciclos y condicionales: Condiciones; if, else; for; do while; break y continue; while.
4. Memoria y arreglos: Heap, stack, code, globals; Notas sobre eficiencia; Arreglos; Direcciones de memoria; Arreglos en funciones; Arreglos en ciclos; Arreglos multidimensionales.
5. Lectura y escritura de archivos: Apertura y modos de apertura; Lectura; Escritura; Archivos binarios.
6. Apuntadores y memoria dinámica: Apuntadores; Operadores \*, & y []; Apuntador NULL; Memoria dinámica; malloc y free; Arreglos con memoria dinámica; Aritmética de apuntadores; Apuntadores y archivos de texto.
7. Cadenas de caracteres: Caracteres especiales y secuencias de escape; Manejo de strings; string.h.
8. Estructuras, uniones: Tipos enum; typedef struct; memoria en la estructura; Funciones que reciben y devuelven estructuras; typedef; Estructuras y apuntadores; Aritmética de apuntadores en estructuras; union; estructuras y uniones; Arreglos de estructuras.
9. Estructuras de datos y recursividad: Listas ligadas; Listas doblemente ligadas; Algoritmos recursivos para (manejo de memoria en) listas ligadas; Algoritmo de Shunting-Yard; Árboles binarios.
10. Clases de almacenamiento: auto, register, static, extern.
11. Apuntadores a funciones: Sintaxis; Funciones que reciben y devuelven apuntadores a funciones; typedef.
12. Algoritmos y complejidad: Invariante de ciclo; Orden; Merge Sort (recursión); Insertion sort.
13. Librerías estándar: string.h; time.h; stdlib.h; stdio.h; streams; stdarg.h; stddef.h; signal.h; setjmp.h; assert.h; math.h; erno.h; locale.h; ctype.h; float.h; limits.h.
14. Posix, pipes y forks: Sistemas Posix; popen; pclose.
15. Señales: signal; raise; fork; psignal.
16. Funciones con número variable de argumentos: va\_list; va\_start,va\_end.
17. Operaciones con bits: Operadores; Generadores de pseudo-aleatorios.
18. Preprocesador: Macros; El preprocesador de C; directivas en gcc; Macros predefinidas; Concatenación; Definición condicional de macros; Compilación condicional.
19. Excepciones: 78 setjmp.h; assert.h.
20. Introducción a la programación orientada a objetos: Conceptos OOP; Objeto; Abstracción; Encapsulamiento; Herencia; Polimorfismo; Sobrecarga de funciones y operadores; Compilación con g++; Tipos de datos; namespace.
21. Clases: class; Datos y funciones miembro; Constructor; Modificadores de acceso; Inicializaciones; Sobrecarga del constructor; Constructor de copia; Destructor; Funciones friend; Funciones inline; Apuntador this; Miembros static.
22. Herencia: Clases base y heredada.
23. Sobrecarga de funciones y operadores: Sobrecarga de funciones; Sobrecarga de operadores; Operadores sobrecargables.
24. Polimorfismo: Polimorfismo de funciones en clases derivadas; Funciones virtuales.
25. Abstracción: Ejemplos de abstracción mediante etiquetas de acceso.
26. Encapsulamiento: Ejemplos de encapsulamiento.
27. Interfaces: Clases abstractas; Funciones puramente virtuales.
28. Temas misceláneos: Makefiles; Multithreading.
 |

|  |  |  |
| --- | --- | --- |
| **ACTIVIDADES DE APRENDIZAJE:** | **RECURSOS MATERIALES Y DIDÁCTICOS:** |  |
| 1. Aprendizaje basado en exposición.
2. Aprendizaje basado en problemas.
3. Discusión grupal.
4. Investigación documental y en línea.
5. Otras sugeridas por el Profesor
 | 1. Pizarrón y gis.
2. Proyector y equipo de audio.
3. Computadora con acceso a internet.
4. Otros sugeridos por el Profesor
 |

|  |  |  |
| --- | --- | --- |
| **PRODUCTOS O EVIDENCIAS DEL APRENDIZAJE:** | **SISTEMA DE EVALUACIÓN: (Sugerido)** |  |
| 1. Tareas.
2. Exámenes.
3. Proyectos.
 | 1. Exámenes
2. Tareas
3. Proyectos

TOTAL 100% |

|  |
| --- |
| **FUENTES DE INFORMACIÓN** |
| **BIBLIOGRÁFICAS\*:** | **OTRAS:** |
| 1. R. Sedgewick. Algorithms in C++. Addison Wesley.
2. B.Preiss. Data Structures and Algorithms with Object Oriented Design Patterns in (C++, Java). http://www.brpreiss.com/
3. C.Cormen, C.Leiserson, R.Rivest y C.Stein. Introduction to Algorithms. MIT Press.
4. J.Kleinberg y E.Tardos. Algorithm Design. Addison Wesley.
5. D.Knuth. The Art of Computer Programming. Vol.1 Fundamental Algorithms, Vol.3 Sorting and Searching. Addison-Wesley.
6. Stallman, Richard M., and Zachary Weinberg. "The C preprocessor." Free Software Foundation (1987).
7. Kernighan BW, Ritchie DM. The C programming language. Englewood Cliffs: PrenticeHall; 1988 Mar 22
8. Goldberg, David. "What every computer scientist should know about floating-point arithmetic." ACM Computing Surveys (CSUR) 23.1 (1991): 5-48.
9. Drepper, Ulrich. "What every programmer should know about memory." Red Hat, Inc 11 (2007): 2007.
10. Mark Allen Weis, Efficient C Programming A practical approach, Prentice Hall.
11. Pitt-Francis, Joe, and Jonathan Whiteley. Guide to scientific computing in C++. Springer Science & Business Media, 2012.
 |  |

\*Citar con formato APA