UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
COORDINACIÓN DE FORMACIÓN BÁSICA
COORDINACIÓN DE FORMACIÓN PROFESIONAL Y VINCULACIÓN UNIVERSITARIA
PROGRAMA DE UNIDAD DE APRENDIZAJE

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: Facultad de ciencias

2. Programa (s) de estudio: Lic. Ciencias computacionales, Lic. Matemáticas Aplicadas, Lic. en Física

3. Vigencia del plan: _____

4. Nombre de la Unidad de Aprendizaje: Introducción a la Programación 5. Clave

6. HC: 2 HL: 3 HT: HPC_____ HCL____ HE: 2 CR___7___

7. Etapa de formación a la que pertenece: Básica

8. Carácter de la Unidad de aprendizaje: Obligatoria _____X_____ Optativa _____

9. Requisitos para cursar la unidad de aprendizaje:

Formuló: Dr. Omar Álvarez Xochihua

Fecha: Agosto de 2016

Vo. Bo. Dr. Alberto Leopoldo Moran y Solares

Cargo: Subdirector

II. PROPÓSITO GENERAL DE LA UNIDAD DE APRENDIZAJE

La unidad de aprendizaje de Introducción a la Programación es de carácter obligatorio dentro de la etapa básica. Su área de conocimiento es Programación, donde fortalece el aprendizaje en lógica programática mediante el uso de un lenguaje de programación. Esta asignatura es teórico práctica y tiene como requisito recomendado haber cursado y aprobado la unidad de aprendizaje de Diseño de Algoritmos.

La finalidad de esta unidad de aprendizaje es capacitar al estudiante en el uso de los fundamentos de programación que le permitan producir programas eficientes que cumplan estándares de calidad, lo cual es requerido por las unidades de aprendizaje de Programación Orientada a Objetos y Estructura de Datos.

III. COMPETENCIA DE LA UNIDAD DE APRENDIZAJE

Desarrollar programas de computadora estructurados, aplicando e integrando los estándares de buenas prácticas y técnicas inherentes a los conceptos de programación, para la adquisición de experiencia en el desarrollo de software, con una actitud crítica y de compromiso.

IV. EVIDENCIA (S) DE DESEMPEÑO

Entrega de un proyecto final que integre las estructuras de control, estructuras de datos y estándares de programación, el cual incluya una propuesta inicial donde se describa su funcionalidad y la programación de actividades. El proyecto puede ser realizado de manera individual o grupal (máximo tres estudiantes) y se deberá presentar semanalmente los avances de la implementación de funcionalidades.

V. DESARROLLO POR UNIDADES

Competencia

Interpretar los conceptos que se involucran en un ambiente de programación de computadoras, mediante la revisión de los paradigmas y fundamentos de programación, para obtener las bases teóricas de la lógica programática, con una actitud crítica.

Contenido **Duración** 3 hrs.

I.- Introducción a la Programación

1. Cronología de los paradigmas de programación
2. Programación estructurada
3. Estructura básica de un programa
4. Edición, compilación y depuración de un programa
5. Estándares de buenas prácticas de programación

Competencia

Aplicar los distintos tipos de datos y expresiones de programación de computadoras, mediante el uso de un lenguaje de programación estructurado, para interpretar la manera de definir y manipular datos dentro de un programa de cómputo, con una actitud crítica.

Contenido **Duración** 3 hrs.

II.- Constantes, variables, tipos, expresiones y asignaciones

1. Tipos de datos constantes y variables
   - 1.1. Definición de variables y constantes (zonas de memoria)
   - 1.2. Tipos de datos: numéricos, carácter, cadena, booleanos
   - 1.3. Operadores de relación, asignación, aritméticos y asignación implícita de tipos
   - 1.4. Jerarquía de operadores
   - 1.5. Operadores de manejo de cadenas
2. Expresiones y sentencias
   - 2.1. Expresiones aritméticas y lógicas
   - 2.2. Sentencias simples y compuestas
   - 2.3. Operadores de incremento y decremento
   - 2.4. Bloque de sentencias

Competencia

Aplicar las estructuras de control en la formulación de programas de computadora, utilizando un lenguaje de programación, para automatizar actividades y procesos de propósito general, con una actitud crítica y creativa.

Contenido **Duración** 15 hrs.

III.- Estructuras de control y depuración de programas: sentencias, condicionales e iteraciones.

1. Fundamentos básicos de estructuras de control
   - 1.1. Operaciones booleanas
   - 1.2. Negación, conjunción, expresiones complejas
   - 1.3. Cálculo de predicados
2. Secuencia
   - 2.1. Análisis de problemas de programación
   - 2.2. Secuencia lógica
   - 2.3. Bloque de sentencias
   - 2.4. Implementación y depuración
3. Selección
   - 3.1. Definición y componentes de una condicional
   - 3.2. Condicionales sencillas
   - 3.3. Condicionales dobles
   - 3.4. Condicionales múltiples
   - 3.5. Condicionales anidadas
   - 3.6. Implementación y depuración
4. Iteración
   - 4.1. Definición y componentes de un ciclo
   - 4.2. Tipos de ciclos: por contador y por centinela
   - 4.3. Anidación de ciclos
   - 4.4. Implementación y depuración

Competencia

Descomponer un programa de computación en secciones, mediante el uso de las diferentes modalidades de funciones, para optimizar la funcionalidad y mantenimiento de código de programación, con una actitud crítica y propositiva.

Contenido **Duración** 3 hrs.

IV.- Funciones

1. Definición y componentes de una función
2. Cuerpo, llamado y prototipos de funciones
3. Funciones con parámetros por valor
4. Funciones con parámetros por referencia
5. Implementación y depuración

Competencia

Diseñar espacios de almacenamiento de datos compuestos, mediante el uso de las diferentes estructuras de datos disponibles en los lenguajes de programación, para optimizar el manejo de memoria y la manipulación de datos dentro de un programa de cómputo, con una actitud creativa y propositiva.

Contenido **Duración** 3 hrs.

V.- Estructuras de datos

1. Estructuras de datos del mismo tipo (arreglos)
   1.1. Arreglos unidimensionales
   1.2. Arreglos multidimensionales
2. Estructuras de datos multi tipo (registros)
3. Implementación y depuración

Competencia

Desarrollar programas de cómputo robustos, a través del uso de funcionalidades avanzadas disponibles en lenguajes de programación, para optimizar el desempeño de los programas de cómputo realizado, con una actitud propositiva y creativo.

Contenido **Duración** 5 hrs.

VI.- Tópicos avanzados de programación

1. Recursividad
   - 1.1. Ejemplos de algoritmos recursivos
   - 1.2. Programación recursiva
   - 1.3. Implementación y depuración
2. Manejo dinámico de memoria
   - 2.1. Definición de memoria dinámica
   - 2.2. Asignación dinámica de memoria
   - 2.3. Uso de memoria dinámica
3. Excepciones
   - 3.1. Tipos de errores y excepciones
   - 3.2. Excepciones no atrapadas
   - 3.3. Implementación y depuración

VI. ESTRUCTURA DE LAS PRÁCTICAS

No. de Práctica	Competencia(s)	Descripción	Material de Apoyo	Duración
1	Aplicar los componentes de un ambiente de desarrollo, mediante el uso de un entorno integrado de desarrollo, para crear y depurar programas de computadora, con iniciativa.	Identificar los elementos básicos de un ambiente integrado de desarrollo (IDE, por sus siglas en inglés)	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	2 hrs.
2	Manipular información en un programa de cómputo, aplicando instrucciones de declaración, entrada y salida de datos, para automatizar el procesamiento de los datos requeridos en un problema, con actitud crítica.	Utilizar las funciones de entrada y salida de datos	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	4 hrs.
3	Formular soluciones programáticas con flujos de datos múltiples, mediante el uso de instrucciones condicionales simples, compuestas, múltiples y anidadas, para generar programas de cómputo que requieran modificar el flujo de ejecución de las instrucciones de un programa, con una actitud creativa y propositiva.	Utilizar y comparar las distintas instrucciones condicionales	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	12 hrs.
4	Formular soluciones programáticas iterativas, mediante el uso de instrucciones de repetición simple, anidada, por contador y por centinela, para generar programas de cómputo que requieran repetir el flujo de ejecución de las instrucciones de un programa, mediante una actitud creativa y propositiva.	Utilizar y comparar las distintas instrucciones de iteración	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	12 hrs.

5	Formular soluciones programáticas modulares, mediante el uso de funciones que reciban parámetros por valor o por referencia, para generar programas de cómputo optimizando su funcionalidad y mantenimiento, con una actitud crítica y propositiva.	Manejo de funciones para optimizar el código de programación generado.	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	6 hrs.
6	Formular soluciones programáticas con datos compuestos, mediante el uso de estructuras de datos disponibles en los lenguajes de programación, para optimizar el manejo de memoria y la manipulación de datos dentro de un programa de cómputo, con una actitud creativa y propositiva.	Utilizar y comparar estructuras de datos mono tipo y multitipo	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	6 hrs.
7	Formular soluciones programáticas robustas, mediante el uso de funcionalidades recursivas, detección y manipulación de errores y manejo de memoria, para optimizar el desempeño de los programas de cómputo realizados, con una actitud propositiva y creativa.	Manejo de elementos avanzados de programación.	Computadora Compilador de lenguaje de programación Ambiente integrado de desarrollo	6 hrs.

VII. METODOLOGÍA DE TRABAJO

Aprendizaje participativo

Durante el desarrollo de la unidad de aprendizaje, el docente fomentará la participación activa de los estudiantes en actividades individuales y grupales. Mediante la discusión de las temáticas vistas en las clases teóricas y la asignación de ejercicios y prácticas en las sesiones de taller y laboratorio. En las actividades grupales el docente asignará un problema a cada equipo, el cual primeramente debe ser analizado en forma individual, posteriormente se discutirá y definirá una solución en equipo previa implementación de la misma. La participación del maestro en la aplicación de esta metodología es de mediador.

Prácticas de laboratorio

En las sesiones de laboratorio, el estudiante llevará a la práctica los conocimientos teóricos vistos en clase y los ejercicios realizados durante el taller, al mismo tiempo que desarrolle habilidades con el manejo de al menos un lenguaje de programación y un ambiente integrado de desarrollo.

Investigación Bibliográfica

Se sugiere solicitar investigación en diferentes fuentes bibliográficas sobre temas de actualidad o temáticas que serán discutidos posteriormente en clase. El propósito de estos trabajos es fomentar el autoaprendizaje y que el estudiante aprenda a realizar investigación en medios electrónicos (Internet), libros, y revistas sobre temas del área. Las fuentes serán tanto en el idioma inglés como español. Los reportes deberán contener las referencias que se utilizaron para la realización del trabajo y debe contar imprescindiblemente una conclusión personal acerca de la investigación. El maestro debe enfatizar a los estudiantes que los reportes escritos sean claros y bien redactados, recalcándoles también las faltas de ortografía.

Ejercicios y exámenes de conocimientos

El maestro deberá aplicar al menos 2 exámenes de conocimientos durante el curso, que permitan identificar la obtención de competencias de los estudiantes. Los exámenes podrán ser de varios tipos, tales como: de preguntas abiertas, opción múltiple y solicitud de programas. Así como, la asignación de ejercicios para ser realizados en la sesión de taller o extra clase, de tal manera que refuercen los conocimientos aprendidos durante la clase teórica. Se solicitará la entrega oportuna y formal de tareas y trabajos de investigación.

VIII. CRITERIOS DE EVALUACIÓN

Criterio de acreditación:

Aplicar al menos 2 exámenes parciales en tiempo y forma.
Cumplir con las prácticas y tareas extra-clase en tiempo y forma.
Cumplir con la presentación del proyecto final en tiempo y forma.
Para la acreditación del curso se atenderá al Estatuto Escolar Vigente, artículos 70-71, por lo que el estudiante deberá contar un mínimo de 80% de asistencias en el periodo. Tener un mínimo aprobatorio de 60 en su calificación final.

Evaluación:

La evaluación general del curso consistirá de exámenes teórico-prácticos, tareas-reportes, prácticas de laboratorio y un proyecto final.
El proyecto final deberá realizarse en equipo de trabajo, la evaluación se dividirá en dos partes el 50% de la calificación será asignado al producto terminado y la segunda se asignará a un reporte escrito y a la exposición oral acerca del proyecto.

Los reportes por escrito será por equipo y los puntos a evaluar son, contenido, claridad y forma, así como ortografía y redacción; para la exposición oral los puntos a evaluar serán, dominio del tema, claridad y estructura. Los alumnos pueden ayudarse en la exposición mediante apoyos visuales tales como proyector de transparencias, acetatos u otros medios.

Calificación:

Los porcentajes de evaluación propuestos serán los siguientes:

Exámenes	30%
Tareas/prácticas	30%
Proyecto final	40 %
Total	**100%**

IX. BIBLIOGRAFÍA

Básica

- Deitel M., Harvey; Paul J., Deitel. (2014). “C++ Cómo programar”. Pearson Educación, 9na. Edición.
- Llopis P, Fernando; Pérez L, Ernesto; Ortuño O, Fernando.(2000). “Introducción a la programación : algoritmos y C/C++”, Digitalia. [clásico]
- Ramírez, Felipe. (2012). “Introducción a la programación: algoritmos y su implementación en VB.Net, C#, Java y C++”. Alfaomega, 2da. Edición.
- http://web.a.ebscohost.com/ehost/detail/detail?vid=2&sid=f83b8bba-1fa8-4c7d-96c9-685cbd0fbdc7%40sessionmgr4003&hid=4112&bdata=Jmxhbmc9ZXMmc2l0ZT1laG9zdC1saXZl#db=e000xww&AN=318031

Complementaria

- Dawson, Michael. (2010). “Python Programming for the Absolute Beginner”, 3rd Edition.
- Joyanes A., Luis. (2015). “Fundamentos de programación: Algoritmos, estructuras de datos y objetos”, McGraw-Hill, 4ta. Edición.

X. PERFIL DEL DOCENTE

El docente de esta asignatura deberá ser un profesionista con formación en el área de computación o áreas afines; con experiencia en docencia y en lógica programática, análisis, diseño e implementación de programas de computadora, y conocimiento de lenguajes de programación basados en el paradigma estructurado.