

## 1. CURSO

CS112. Programación I (Obligatorio)

## 2. INFORMACIÓN GENERAL

2.1 Curso	:	CS112. Programación I
2.2 Semestre	:	2 <sup>do</sup> Semestre.
2.3 Créditos	:	4
2.4 horas	:	3 HT; 3 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	CS100. Fundamentos de Computación. (1 <sup>er</sup> Sem) CS100. Fundamentos de Computación. (1 <sup>er</sup> Sem)

## 3. PROFESORES

Atención previa coordinación con el profesor

## 4. INTRODUCCIÓN AL CURSO

Este es el segundo curso en la secuencia de los cursos introductorios a la Ciencia de la Computación. El curso introducirá a los participantes en los diversos temas del área de computación como: algoritmos, estructuras de datos, ingeniería del software, etc.

## 5. OBJETIVOS

- Introducir al alumno a los fundamentos del paradigma de orientación a objetos, permitiendo asimilar los conceptos necesarios para desarrollar sistemas de información.

## 6. RESULTADOS DEL ESTUDIANTE

- 1) Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Evaluar**)
- 2) Diseñar, implementar y evaluar una solución basada en computación para cumplir con un conjunto determinado de requisitos computacionales en el contexto de las disciplinas del programa. (**Evaluar**)
- 5) Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas a la disciplina del programa. (**Familiarizarse**)
- 6) Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. (**Usar**)

## 7. TEMAS

Unidad 1: Visión General de los Lenguajes de Programación (1)	
Resultados esperados: 1	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>• Breve revisión de los paradigmas de programación.</li> <li>• Comparación entre programación funcional y programación imperativa.</li> <li>• Historia de los lenguajes de programación.</li> </ul>	<ul style="list-style-type: none"> <li>• Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]</li> </ul>
Lecturas : [Stroustrup2013], [Deitel17]	

<b>Unidad 2: Máquinas virtuales (2)</b>	
<b>Resultados esperados: 1,6</b>	
<b>Temas</b>	<b>Objetivos de Aprendizaje</b>
<ul style="list-style-type: none"> <li>• El concepto de máquina virtual.</li> <li>• Tipos de virtualización (incluyendo Hardware / Software, OS, Servidor, Servicio, Red) .</li> <li>• Lenguajes intermedios.</li> </ul>	<ul style="list-style-type: none"> <li>• Explicar el concepto de memoria virtual y la forma cómo se realiza en hardware y software [Familiarizarse]</li> <li>• Diferenciar emulación y el aislamiento [Familiarizarse]</li> <li>• Evaluar virtualización de compensaciones [Evaluar]</li> </ul>
<b>Lecturas : [Stroustrup2013], [Deitel17]</b>	

Unidad 3: Sistemas de tipos básicos (6)	
Resultados esperados: 1,6	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>• Tipos como conjunto de valores junto con un conjunto de operaciones. <ul style="list-style-type: none"> <li>– Tipos primitivos (p.e. números, booleanos)</li> <li>– Composición de tipos contruídos de otros tipos (p.e., registros, uniones, arreglos, listas, funciones, referencias)</li> </ul> </li> <li>• Declaración de modelos (enlace, visibilidad, alcance y tiempo de vida).</li> <li>• Vista general del chequeo de tipos.</li> </ul>	<ul style="list-style-type: none"> <li>• Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse]</li> <li>• Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse]</li> <li>• Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse]</li> <li>• Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar]</li> <li>• Dar un ejemplo de un programa que no verifique tipos en un lenguaje particular y sin embargo no tenga error cuando es ejecutado [Familiarizarse]</li> <li>• Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar]</li> <li>• Explicar como las reglas de tipificación definen el conjunto de operaciones que legales para un tipo [Familiarizarse]</li> <li>• Escribir las reglas de tipo que rigen el uso de un particular tipo compuesto [Usar]</li> <li>• Explicar por qué indecidibilidad requiere sistemas de tipo para conservadoramente aproximar el comportamiento de un programa [Familiarizarse]</li> <li>• Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar]</li> <li>• Discutir las diferencias entre, genéricos (<i>generics</i>), subtipo y sobrecarga [Familiarizarse]</li> <li>• Explicar múltiples beneficios y limitaciones de tipificación estática en escritura, mantenimiento y depuración de un software [Familiarizarse]</li> </ul>
Lecturas : [Stroustrup2013], [Deitel17]	

<b>Unidad 4: Conceptos Fundamentales de Programación (10)</b>	
<b>Resultados esperados: 1,6</b>	
<b>Temas</b>	<b>Objetivos de Aprendizaje</b>
<ul style="list-style-type: none"> <li>● Diseño orientado a objetos: <ul style="list-style-type: none"> <li>– Descomposición en objetos que almacenan estados y poseen comportamiento</li> <li>– Diseño basado en jerarquía de clases para modelamiento</li> </ul> </li> <li>● Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse]</li> <li>● Variables y tipos de datos primitivos (ej., números, caracteres, booleanos)</li> <li>● Expresiones y asignaciones.</li> <li>● Estructuras de control condicional e iterativas.</li> </ul>	<ul style="list-style-type: none"> <li>● Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar]</li> <li>● Identifica y describe el uso de tipos de datos primitivos [Familiarizarse]</li> <li>● Escribe programas que usan tipos de datos primitivos [Usar]</li> <li>● Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar]</li> <li>● Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar]</li> <li>● Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar]</li> </ul>
<b>Lecturas :</b> [Stroustrup2013], [Deitel17]	

<b>Unidad 5: Funciones (3)</b>	
<b>Resultados esperados: 1,6</b>	
<b>Temas</b>	<b>Objetivos de Aprendizaje</b>
<ul style="list-style-type: none"> <li>● Paso de funciones y parámetros.</li> <li>● Paso de parámetros</li> <li>● Sobrecarga en funciones</li> <li>● Fundamentos de la recursividad</li> <li>● Conceptos de plantillas en funciones</li> </ul>	<ul style="list-style-type: none"> <li>● Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar]</li> <li>● Entiende y aplica el concepto de paso de parámetros a una función, tanto por valor como por referencia.[Usar]</li> <li>● Identifica y aplica el concepto de sobrecarga de funciones.[Usar]</li> <li>● Describe el concepto de recursividad y da ejemplos de su uso [Familiarizarse]</li> <li>● Diseña, implementa y aplica el concepto de plantillas asociado a la necesidad de crear funciones genéricas.[Usar]</li> </ul>
<b>Lecturas :</b> [Stroustrup2013], [Deitel17]	

Unidad 6: Arreglos y Punteros (3)	
Resultados esperados: 1,6	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>• Definición de arreglos</li> <li>• Arreglos multidimensionales</li> <li>• Fundamentos sobre punteros</li> <li>• Administración dinámica de memoria</li> <li>• Conceptos avanzados de Punteros</li> </ul>	<ul style="list-style-type: none"> <li>• Entiende e implementa arreglos unidimensionales. [Familiarizarse]</li> <li>• Diseña y aplica el concepto de arreglos multidimensionales.[Usar]</li> <li>• Entiende y aplica el concepto de referencias y punteros.[Familiarizarse]</li> <li>• Entiende, aplica y evalúa la relación entre punteros y arreglos.[Evaluar]</li> <li>• Entiende e implementa la gestión dinámica de la memoria. Diferenciando las regiones de memoria: heap y stack. [Evaluar]</li> <li>• Diseña, implementa y evalúa el concepto de puntero a puntero, puntero a función, entre otros conceptos.[Evaluar]</li> </ul>
Lecturas : [Stroustrup2013], [Deitel17]	

Unidad 7: Programación orientada a objetos (2)	
Resultados esperados: 1,6	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>● Diseño orientado a objetos: <ul style="list-style-type: none"> <li>– Descomposición en objetos que almacenan estados y poseen comportamiento</li> <li>– Diseño basado en jerarquía de clases para modelamiento</li> </ul> </li> <li>● Lenguajes orientados a objetos para la encapsulación: <ul style="list-style-type: none"> <li>– privacidad y la visibilidad de miembros de la clase</li> <li>– Interfaces revelan único método de firmas</li> <li>– clases base abstractas</li> </ul> </li> <li>● Definición de las categorías, campos, métodos y constructores.</li> <li>● Las subclases, herencia y método de alteración temporal.</li> <li>● Subtipificación: <ul style="list-style-type: none"> <li>– Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos.</li> <li>– Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos.</li> <li>– Relación entre subtipos y la herencia.</li> </ul> </li> <li>● Uso de colección de clases, iteradores, y otros componentes de la librería estándar.</li> <li>● Asignación dinámica: definición de método de llamada.</li> </ul>	<ul style="list-style-type: none"> <li>● Diseñar e implementar una clase [Usar]</li> <li>● Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar]</li> <li>● Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar]</li> <li>● Comparar y contrastar (1) el enfoque proceduracional/funcional- definiendo una función por cada operación con el uso de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar]</li> <li>● Explicar la relación entre la herencia orientada a objetos (código compartido y <i>overriding</i>) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse]</li> <li>● Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar]</li> <li>● Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, seleccionar la forma más natural por cada lenguaje [Usar]</li> </ul>
Lecturas : [Stroustrup2013], [Deitel17]	

Unidad 8: Plantillas y STL (2)	
Resultados esperados: 1,6	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>● Definición de plantillas en clases</li> <li>● Conceptos básicos sobre la Standard Template Library (STL)</li> </ul>	<ul style="list-style-type: none"> <li>● Entiende los conceptos de plantillas en clases. [Familiarizarse]</li> <li>● Implementa y crea nuevos tipos de datos genéricos. [Usar]</li> <li>● Entiende las estructuras básicas de la STL. [Familiarizarse]</li> <li>● Usa las estructuras de datos básicas como: pila, cola, lista, vector contenidos en la STL. [Usar]</li> </ul>
Lecturas : [Stroustrup2013], [Deitel2017]	

Unidad 9: Conceptos Avanzados (2)	
Resultados esperados: 1,6	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>• Definición de sobrecarga de operadores</li> <li>• Manipulación de entrada y salida de datos (I/O)</li> <li>• Patrones de diseño</li> </ul>	<ul style="list-style-type: none"> <li>• Entiende los conceptos de sobrecarga de operadores. [Familiarizarse]</li> <li>• Implementa la sobrecarga de operadores permitidos en el lenguaje de programación. [Usar]</li> <li>• Entiende los conceptos de manipulación de archivos. [Familiarizarse]</li> <li>• Crea programas de lectura y escrita en archivos. [Usar]</li> <li>• Entiende los conceptos de patrones de diseño. [Familiarizarse]</li> </ul>
Lecturas : [Stroustrup2013], [Deitel2017]	

## 8. PLAN DE TRABAJO

### 8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

### 8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

### 8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

## 9. SISTEMA DE EVALUACIÓN

\*\*\*\*\* EVALUATION MISSING \*\*\*\*\*

## 10. BIBLIOGRAFÍA BÁSICA