**[DECC | PUJC](http://decc.javerianacali.edu.co/decc/)**

Departamento de Electrónica y Ciencias de la Computación



**Programación Orientada a Objetos**

**Información Básica**

Créditos: 3



Horas de trabajo acompañado: 5 horas por semana ( 3 horas de clase, 2 horas de taller)





Horas de trabajo independiente: 3 horas por semana



Pre-requisitos: 300CIP002 Fundamentos y estructuras de programación



Tipo de curso: Núcleo de Formación Fundamental.



**Descripción del Curso**

Este curso presenta conceptos de diseño y programación orientada a objetos (OO). El curso utiliza el enfoque "Los objetos primero", que es la base del paradigma OO. Adicionalmente este curso introduce nociones de ingeniería de software como los diagramas de clase UML, buenas prácticas de diseño, las pruebas unitarias y el manejo de excepciones.

**Objetivo de la asignatura:** Analizar, diseñar e implementar soluciones de problemas de mediana escala mediante la aplicación del paradigma de la programación orientada a objetos y prácticas de ingeniería de software.

* Objetivo del capítulo 1: Programación orientada a objetos

Identificar las características de la programación orientada a objetos mediante el estudio de técnicas para representar y codificar clases, objetos, métodos, atributos con las notaciones UML y el lenguaje C++

* Objetivo del capítulo 2: Diseño orientado a objetos

Diseñar e implementar soluciones a problemas usando el paradigma orientado a objetos a partir la aplicación de prácticas que favorezcan la abstracción, encapsulamiento, herencia y polimorfismo en lenguaje C++.

* Objetivo del capítulo 3: Desarrollo de software a media escala

Reconocer técnicas de desarrollo de software de media escala mediante el estudio de técnicas de gestión de excepciones, pruebas unitarias, principios de diseño de software y patrones de diseño propuestos en la ingeniería de software para construir productos de software de calidad.

**Competencias técnicas específicas que se desarrollan**

* Diseño de diagramas de clases con UML (Avanzado)
* Programación en C++ (Intermedio)
* Stack para gestión de versiones [Git – GITHUB] (Básico)
* Uso de línea de comandos (Básico)
* Elaboración de archivos Makefiles
* Interacción con IDE variados (Visual studio code, CLion)

**Contenido**

**Capítulo 1: Paradigma de programación orientada a objetos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sesión** | **Horas teóricas** | **Prácticas acompañadas** | **Temas** | **Profundidad** | **Bibliografía** |
| 1 | 2 | 0 | Presentación del curso, profesor y estudiantes  Ejercicios de programación de repaso | Evaluación |  |
| 2 | 2 | 1 | Introducción al modelo Orientado a objetos y comparación con la programación estructurada  Introducción a C++ | Familiaridad | [1] Capítulo 2  [4] Sección 1.4 - 1.5 |
| 3 - 5 | 4 | 3 | *Elementos principales de la programación orientada a objetos*.  Clases y métodos, atributos primitivos y compuestos  Diferencia atributos vs variables locales, Métodos, Getters | setters  Modificadores de acceso  Constructores por defecto y con parámetros, instanciación de objetos. Práctica en C++ | Evaluación | [5] Capítulo 3. Sección 3.1 -3.3 |
| 6 | 3 |  | UML diagramas de clases  Relaciones entre clases asociación, agregación y dependencia |  | [1] Capítulo 11  [4] Capítulo 16 |
| 7 | 1 | 1 | Separación de clases .C y .CPP  Makefiles y compilación de múltiples archivos  Guardas | Evaluación | [3] Sección 16.3 |
| 8 | 2 | 1 | Clases contenedoras: list, sets, maps  Iteradores | Evaluación | [6] Capítulo 20  [1] Capítulo 18 |
| 9 | 2 | 0 | Modelado de un sistema: técnicas para identificar clases, atributos y métodos. | Evaluación | [4] Capítulo 4. |
| 10 -11 | 0 | 5 | Ejercicio diseño, codificación y mejora problema orientado a objetos en C++ | Evaluación | [3] Capitulo 16 y 17 |
| 12 -13 |  | 5 | Acompañamiento proyecto práctico y sustentación |  |  |

**Total de horas:** 32

**Capítulo 2: Diseño orientado a objetos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sesión** | **Horas teóricas** | **Prácticas acompañadas** | **Temas** | **Profundidad** | **Bibliografía** |
| 14 -15 | 3 | 1 | Introducción pilares de la programación orientada a objetos: abstracción, encapsulamiento, herencia y polimorfismo  Herencia. Motivación para hacer herencia  Conceptos claves: subclases y superclase  Representación en UML  Herencia de implementación: cuando subclase usa, modifica o remplaza valores o funcionalidades.  Modificador de acceso protegido  Llamado a constructores  Diferencias Herencia / Asociación/Agregación  Usos correctos e incorrectos de la herencia. Regla *is-a* y principio de sustitución.  Ejercicios en C++ | Evaluación | [1] Capítulo 11  [2] Capítulo 16  [4] Capítulo 17 |
| 15 - 16 | 2 | 2 | Memoria dinámica en C++  Stack, Heap  Paso de parámetros por referencia valor y apuntadores  Memory leak  Dangling pointers  Destructores  Ejercicios C++ | Evaluación | [2] Capítulo 17  [6] Capítulo 17 |
| 17 -18 | 3 | 2 | Polimorfismo: sobrecarga y sobreescritura  Casteo implícito y explícito  Ejercicios C++ | Evaluación | [1] Capítulo 12  [5] Capítulo 3. Sección 3.4 |
| 19 | 2 |  | Clases y métodos **estáticos** y **const** | Evaluación | [3] Capítulo 18 |
| 20 | 1 | 2 | Técnicas de abstracción: clases abstractas, herencia múltiple e interfaces, buenas prácticas de herencia  Ejercicios C++: métodos virtuales, enlace dinámico | Evaluación | [1] Capítulo 12  [5] Capítulo 3. Sección 3.4 |
| 21 | 2 | 0 | Estándares de codificación. Importancia de los estándares de codificación, revisión de estándar de codificación de Google para C++  Revisión de buenas prácticas de codificación: nombramiento de variables, organización de código, nombramiento de métodos. | Uso | [7] |
| 22 | 0 | 3 | Acompañamiento en proyecto práctico y sustentación | Evaluación |  |

**Total de horas:** 23

**Capítulo 3: Desarrollo de software a media escala**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sesión** | **Horas teóricas** | **Prácticas acompañadas** | **Temas** | **Profundidad** | **Bibliografía** |
| 23 | 2 |  | Manejo de excepciones en C++ | Uso | [1] Capítulo 14 |
| 24 | 2 | 1 | Pruebas unitarias en C++ | Uso | [2] Capítulo 19  [6] Capítulo 26 |
| 25-26 | 3 | 2 | Principios GRASP en el diseño orientado a objetos | Uso | [4] Capítulo 17 y Capítulo 18 |
| 27 -28 | 3 | 2 | Patrones de diseño de creación, de comportamiento y estructurales | Familiaridad | [4] Capítulo 25 |
| 29 -30 | 2 | 2 | Diseño detallado vs diseño de alto nivel - Patrones de arquitectura | Familiaridad | [4] Capítulo 33 |
| 30 -32 | 0 | 6 | Acompañamiento en proyecto práctico y sustentación | Evaluación |  |

**Total de horas:** 25

**Integración Curricular**

**Resultados de Programa (ABET)**

Los graduados del programa tendrán la capacidad de:

1. Analizar un problema computacional complejo y aplicar los principios de la computación y otras disciplinas relevantes para identificar soluciones
2. Diseñar, implementar y evaluar una solución basada en la computación para satisfacer un conjunto dado de requisitos informáticos en el contexto de la disciplina del programa. La habilidad para diseñar, implementar y evaluar procesos y sistemas computacionales.
3. Comunicarse eficazmente en una variedad de contextos profesionales.
4. Reconocer las responsabilidades profesionales y hacer juicios informados en la práctica de la computación basados en principios legales y éticos.
5. Funcionar efectivamente como miembro o líder de un equipo dedicado a actividades propias de la disciplina del programa.
6. Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en la computación

**Relevancia del curso con los resultados de programa**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Resultados de Programa | | | | | | |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| Relevancia | 2 | 4 |  |  | 1 | 4 |

Escala: (1) baja relevancia - (5) alta relevancia.

**Calificación y balance de evaluación**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instrumento de Evaluación** | **%** | **Competencias** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| **Proyecto 1** | 20% | 4% | 8% |  |  | 2% |  |
| **Proyecto 2** | 20% | 4% | 8% |  |  | 2% | 6% |
| **Proyecto 3** | 20% | 4% | 8% |  |  | 2% | 3% |
| **Ensayo** | 10% |  |  |  |  | 3% |  |
| **Tareas, participación** | 30% | 12% | 12% |  |  |  | 12% |
|  | **100%** | **18%** | **36%** | **0%** | **0%** | **9%** | **37%** |

**Relación de objetivos del curso con resultados de programa y estrategia de evaluación**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resultados del Programa** | **Indicadores de Desempeño** | **Objetivos/Contenido del Curso** | **Actividades de Aprendizaje** | **Instrumentos de Medición** |
| 1. Analizar un problema computacional complejo y aplicar los principios de la computación y otras disciplinas relevantes para identificar soluciones | 1.1. Identificar los principios científicos y de ingeniería fundamentales que rigen un proceso o sistema determinado y relacionar los conceptos y principios teóricos con la resolución de problemas prácticos (Conocimiento - Análisis - Síntesis)  1.2. Resolver problemas relacionados con la disciplina y otras áreas utilizando conocimientos, modelos y formalismos propios de la computación. (Aplicación)  1.3. Analizar conjuntos de datos. (Análisis)  1.4. Interpretar los resultados de modelos matemáticos para estimar la precisión y la fiabilidad. (Comprensión).  1.5. Identificar componentes, interacciones, relaciones e interfaces entre componentes. Sintetizar la información, las pruebas y los hechos necesarios para analizar un problema y formular una hipótesis (Análisis - Síntesis) | Todos | Sesiones magistrales, ejercicios en clase, proyecto, talleres, tareas, lecturas, socialización de trabajos | Proyecto 1  Proyecto 2  Proyecto 3  Tareas y participación en clase |
| 2. Diseñar, implementar y evaluar una solución basada en la computación para satisfacer un conjunto dado de requisitos informáticos en el contexto de la disciplina del programa. La habilidad para diseñar, implementar y evaluar procesos y sistemas computacionales. | 2.1. Utilizar herramientas de desarrollo de software. Utilizar estándares de codificación al implementar componentes de software. (Aplicación).  2.2. Diseñar procesos y artefactos de software utilizando técnicas, herramientas y notación adecuadas. Implementar e integrar componentes de software que sigan fielmente los criterios de diseño. (Síntesis)  2.3. Evaluar un componente de software con respecto a las complejidades temporales y espaciales. Evaluar y verificar soluciones de software con respecto a los requisitos y restricciones del sistema. (Evaluación).  2.4. Describir procesos de forma declarativa, abstrayéndose del comportamiento de la implementación. (Comprensión)  2.5. Utilizar las matemáticas, la lógica y el lenguaje de ingeniería para especificar los requisitos funcionales y no funcionales y las propiedades de los sistemas y procesos. (Aplicación) | Todos | Sesiones magistrales, ejercicios en clase, proyecto, talleres, tareas, lecturas, socialización de trabajos | Proyecto 1  Proyecto 2  Proyecto 3  Tareas y participación en clase |
| 3. Comunicarse eficazmente en una variedad de contextos profesionales. | 3.1. Producir una comunicación escrita eficaz en cuanto a: estructura, coherencia, fluidez, ortografía, puntuación y gramática correctas (Aplicación).  3.2. Comunicar eficazmente con un lenguaje, un estilo, un ritmo, una fluidez y unas estrategias no verbales adecuadas a la audiencia a la que se dirige (Aplicación).  3.3. Utilizar recursos gráficos para comunicar y explicar una idea (Aplicación).  3.4. Defender ideas con precisión y claridad (Evaluación). |  |  |  |
| 4. Reconocer las responsabilidades profesionales y hacer juicios informados en la práctica de la computación basados en principios legales y éticos. | 4.1. Identificar los códigos éticos relacionados con la disciplina (Conocimiento).  4.2. Mostrar responsabilidad y comportamiento profesional. Reconocer la importancia de la profundidad y la amplitud de los conocimientos. (Valoración - Comprensión).  4.3. Analizar el impacto local y global de la computación y la ingeniería. Identificar los pros y los contras en las decisiones éticas relacionadas principalmente con los comportamientos profesionales. Juzgar los impactos de la computación y la ingeniería en el mundo. Argumentar y justificar las decisiones éticas (Análisis - Evaluación).  4.4. Capacidad para analizar el impacto local y global de la computación en los individuos, las organizaciones y la sociedad. Identificar los acontecimientos mundiales a los que probablemente afecta la computación y la ingeniería (Comprensión).  4.5. Utilizar los conocimientos para identificar los impactos de la computación y la ingeniería en una solución dada (Aplicación). |  |  |  |
| 5. Funcionar efectivamente como miembro o líder de un equipo dedicado a actividades propias de la disciplina del programa. | 5.1. Reconocer el papel cada vez más importante de la computación en entornos multidisciplinares (Conocimiento).  5.2. Integrar puntos de vista, información, retroalimentación y críticas para proponer una solución (Síntesis).  5.3. Dirigir y gestionar las tareas del equipo y tomar decisiones basadas en la información del equipo (Respuesta)  5.4. Seguir los calendarios y adaptar los recursos para cumplir los hitos (Aplicación). | Todos | Escritura de un ensayo argumentativo  Proyectos | Ensayo  Proyecto 1  Proyecto 2  Proyecto 3  Tareas y actividades en clase |
| 6. Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en la computación | 6.1. Utilizar herramientas de diseño, simulación y modelado de software. Reconocer la importancia de la modelización en la solución de un problema (Aplicación - Comprensión).  6.2. Combinar herramientas de software y hardware para resolver un problema (Síntesis).  6.3. Mostrar flexibilidad para adaptarse a diferentes lenguajes y paradigmas de programación (Valoración).  6.4. Evaluar las decisiones de diseño basándose en principios matemáticos y de computación.  6.5. Aplicar nuevos conocimientos para resolver un problema o desarrollar una solución (Aplicación).  6.6. Interpretar y evaluar la información procedente de múltiples fuentes y relacionarla con conocimientos previos (Síntesis - Evaluación). | Todos | Escritura de un ensayo argumentativo  Proyectos  Actividades en clase | Ensayo  Proyecto 1  Proyecto 2  Proyecto 3  Tareas y actividades en clase |

**Bibliografía**

1. C++ programming: an object-oriented approach. Forouzan, B. A., & Gilberg, R. F, 2019.
2. Rai, L. (2015). Programming in C++ (Vol. 4).
3. C how to program, Sixth Edition; Deitel, Ed. Deitel, P., & Deitel, H. 2010.
4. Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition.
5. Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Conallen, J., & Houston, K. A. (2007). Object-Oriented Analysis and Desing wtih Applications (Third). Addison-Wesley.
6. Stroustrup, B. (2014). Programming - Principles and Practice Using C++. In Addison-Wesley.
7. Google (2021) C++ style guide. https://google.github.io/styleguide/cppguide.html

**Instalaciones**

Salón de clase con computador y proyector. Laboratorio de Ingeniería de Sistemas y Computación.

**Material de este semestre**

