Programación 2

1. Características generales

Nombre:	Programación 2
Sigla:	CI-0113
Créditos:	4
Horas lectivas:	5
Requisitos:	CI-0112 Programación 1
Correquisitos:	-
Clasificación:	Curso propio
Ciclo de carrera:	I ciclo, 2do año
Docente(s):	Enrique Muñoz Monge
Datos de contacto:	alejandro.munozmonge@ucr.ac.cr, Oficina 237.
Grupo:	02-05
Semestre y año:	II-2024
Horario y lugar de clases:	02 L: 7-9:50 / J: 7-8:50; 304-IF
	05 L: 16-17:50 / J: 15-17:50; 201-IF
Horario y lugar de consulta:	K: 9-12:00 / J: 13-15:00; Oficina 237
Asistente:	Pendiente

2. Descripción del curso

En este curso el estudiante extiende su conocimiento en la programación de computadoras aprendiendo técnicas de abstracción que le ayuden a construir software de mayor complejidad y calidad. Las técnicas de abstracción son dependientes del lenguaje de programación, por ejemplo, las plantillas y el polimorfismo. El curso se concentra en técnicas de abstracción para construir software reutilizable y genérico. Se entiende por software reutilizable aquel que puede ser usado en diversos contextos, por ejemplo, las funciones de biblioteca. Se entiende por software genérico al software reutilizable que además puede utilizarse con tipos de datos arbitrarios, por ejemplo, un contenedor. En el curso el estudiante aprende tanto a reutilizar software como a crear software reutilizable y genérico.

3. Objetivos

Objetivo general

El objetivo general del curso es que el estudiante aprenda a resolver problemas de programación aplicando técnicas de abstracción para la creación y uso de software genérico y reutilizable.

Objetivos específicos

Durante este curso cada estudiante desarrollará habilidades para:

1. Explicar el modelo de ejecución del lenguaje de programación (máquina nocional) para implementar programas de forma correcta.
2. Aplicar diversas técnicas de abstracción para crear software genérico y reutilizable.
3. Utilizar software genérico para resolver problemas de forma eficiente.
4. Generar y aplicar pruebas unitarias a software genérico.
5. Aplicar buenas prácticas de programación (por ejemplo, documentación de código, programación defensiva y manejo de excepciones).

4. Contenidos

Objetivo específico	Eje temático	Desglose

1	Generalidades	a. Descripción del uso de la memoria estática, memoria automática y la memoria dinámica en el lenguaje de programación a usar en el curso. b. Descripción y comparación de los tipos de memoria disponibles en el lenguaje de programación del curso. c. Descripción de la estructura de proyecto en los ambientes de programación a usar en el curso. d. Descripción del depurador disponible en al menos uno de los ambientes de programación a usar en el curso. e. Descripción de los distintos mecanismos disponibles en el lenguaje de programación a usar en el curso para pasar a los métodos de una clase datos de tipos preconstruidos y objetos. f. Definición de métodos de instancia y métodos de clase en el lenguaje de programación a usar en el curso.
2	Especificación	a. Pautas para la especificación de clases concretas, clases abstractas y clases parametrizadas.
2	Herencia y polimorfismo	a. Definición de clases abstractas. b. Definición de clases que derivan de otras, ya sean abstractas o concretas. c. Definición de tipos polimórficos. d. Definición de métodos (y operadores, si el lenguaje lo permite) polimórficos.

3	Parametrización	a. Mecanismos de abstracción provistos por el lenguaje de programación para crear código genérico. b. Descripción general de una biblioteca de clases parametrizadas de uso estandarizado en el lenguaje de programación a usar en el curso. c. Descripción del uso de algunas clases parametrizadas de la biblioteca referida anteriormente, de acuerdo con las necesidades de los proyectos de programación y los ejemplos desarrollados en el curso.
2, 3	Estructuras de datos y algoritmos	a. Implementación de al menos una estructura de datos compleja, mediante asignación dinámica de memoria, y sus algoritmos.
5	Manejo de excepciones	a. Definición de clases de excepciones. b. Descripción del levantamiento de excepciones. c. Descripción de la captura y tratamiento de excepciones.
2, 3	Manejo de archivos planos	a. Lectura y escritura de archivos de texto y binarios. b. Manejo de archivos secuenciales y de acceso aleatorio.
4	Pruebas de programas	a. Pruebas de constructores y destructores. b. Pruebas de métodos modificadores. c. Pruebas de métodos observadores. d. Ordenamiento de los tipos de pruebas en un controlador de pruebas.
2	Bibliotecas	Creación y utilización de bibliotecas estáticas (ej: .lib, .a) y dinámicas (ej: .dll, .so) como medios de distribución de código reutilizable.

	Temas opcionales (a convenir entre el profesor y los estudiantes)	a. Programación de interfaces gráficas de usuario (GUI). b. Programación de videojuegos. c. Programación orientada a eventos. d. Programación de bases de datos (ej: SQLite). e. Programación procedimental (ej: C) f. Sobrecarga de operadores (ej: clase Fracción, String) g. Control de versiones (ej: Git, Subversion) h. Herramientas de generación de código (ej: compilador, linker, makefiles) i. Programación de expresiones regulares para procesamiento de texto

5. Metodología

Para alcanzar los objetivos del curso se seguirá una metodología híbrida constructivista-tradicional. El estudiante dedicará 5 horas a la semana a actividades presenciales y 7 horas a actividades extra-clase. Las 5 horas presenciales pueden combinarse entre lecciones magistrales en aula y acompañamiento del docente en un laboratorio de computadoras. En las 7 horas extra-clase el estudiante resolverá problemas de programación individuales y proyectos en parejas.

Las sesiones de consulta podrán ser a través de cita previa solicitada por el estudiante. Los materiales usados en el curso, enunciados de las evaluaciones y entrega de estas serán a través del aula virtual alojada en Mediación Virtual. También se usará la plataforma de git de la UCR para la entrega y seguimiento del progreso en las tareas programadas del curso.

6. Evaluación

%	Rubro	Descripción	Fecha*
25	Examen 1	Examen individual, presencial, en papel.	Grupo 02: 7 de octubre Grupo 05: 10 de octubre
30	Examen 2	Examen individual, presencial, en papel.	Grupo 02: 25 de noviembre Grupo 05: 28 de noviembre

15	Trabajos cortos	Exámenes cortos (quices) Trabajos en clase Problemas en juez en línea Problemas inventados	
15	Proyecto 1	Proyecto programación	4 de octubre
15	Proyecto 2	Proyecto programación	22 de noviembre

El profesor planeará **problemas a resolver** a los estudiantes. Los problemas están organizados en secciones temáticas, aproximadamente una cada semana y media. Algunos problemas requerirán más detalles o conceptos de los vistos en clase, con el fin de que la persona estudiante desarrolle habilidades autodidácticas requeridas en el ejercicio de la profesión.

Los estudiantes resolverán problemas de programación siguiendo la metodología y usando herramientas de trabajo por **proyectos**. Los proyectos deben resolverse en equipos de estudiantes que el número de integrantes pueden variar entre proyectos. Cada proyecto se evaluará mediante al menos un avance, en un repositorio de control de versiones y con presentaciones de producto.

Se realizarán dos **exámenes** tras cubrir los temas correspondientes indicados en la tabla. El material cubierto en cada examen es acumulativo del previo por la naturaleza de los contenidos. A menos de que se indique lo contrario, los exámenes serán realizados en papel, ya sea en horario de clase, un miércoles o un sábado, a convenir en función del avance del grupo, y durante un período de tres horas. Por la naturaleza acumulativa de los contenidos, los exámenes abarcan los contenidos del anterior acumulativamente.

7. Cronograma

Los temas del curso se presentan enmarcados a un paradigma de programación de acuerdo con siguiente cronograma. Las columnas de la tabla indican el número de tema, el paradigma y los conceptos más relevantes, el eje temático (sección 4 de contenidos) asociado, las actividades realizadas para cumplir con los objetivos, la duración en semanas (S), y las fechas tentativas.

#	Paradigma y conceptos	Eje temático	Fechas
1	**Generalidades** Tipos de memoria: estática, automática y dinámica. Estructura de proyectos en ambientes de programación, depurador. Métodos de instancia y métodos de clase. Pasar a los métodos de una clase datos de tipos preconstruidos y objetos.	Generalidades. Especificación.	12-agos 30-agos
2	**Programación orientada a objetos (OOP)** Pautas para la especificación de clases concretas, clases abstractas y clases parametrizadas. Manejo de archivos de texto y binarios, secuenciales y de acceso aleatorio. Pruebas de constructores y destructores, métodos modificadores, métodos observadores. Ordenamiento de los tipos de pruebas en un controlador de pruebas.	Especificación. Manejo de archivos planos. Pruebas de programas.	2-sep 20-sep
3	**OOP: Herencia y polimorfismo** Clases abstractas. Clases que derivan de otras, ya sean abstractas o concretas. Tipos polimórficos. Métodos y operadores polimórficos.	Herencia y polimorfismo.	23-sep 11-oct
4	**Programación genérica** Mecanismos de abstracción para crear código genérico. Biblioteca de clases parametrizadas de uso estandarizado. Ejemplos de clases parametrizadas haciendo uso de bibliotecas de clases parametrizadas. Clases de excepciones. Levantamiento de excepciones. Captura y tratamiento de excepciones.	Parametrización Manejo de excepciones.	14-oct 28-oct
5	**Programación orientada a objetos genérica** Implementación de estructuras de datos complejas, mediante asignación dinámica de memoria, y sus algoritmos. Bibliotecas estáticas (ej: .lib, .a) y dinámicas (ej: .dll, .so) como medios de distribución de código reutilizable.	Estructuras de datos y algoritmos. Bibliotecas.	31-oct 14-nov
6	**Temas opcionales** Interfaces gráficas de usuario. Videojuegos. Programación orientada a eventos. Bases de datos. Programación procedimental en C. Sobrecarga de operadores. Control de versiones Git. Herramientas de generación de código. Expresiones regulares para procesamiento de texto.	Temas opcionales.	15-nov 22-novl

8. Bibliografía

Libro de texto recomendado:

1. Stroustrup, Bjarne; "The C++ Programming Language"; 3rd edition; Addison-Wesley; 1998. http://www.research.att.com/~bs/3rd.html
2. Deitel, Harvey M.; Deitel, Paul J.; "C++ How to Program, 9/e"; Prentice-Hall; 2015.

Otra bibliografía de apoyo:

1. Osherove, Roy. “The art of Unit Testing”; 2nda edición; Manning Pub; 2014.
2. Myers, Glenford J.; "The art of software testing 2nd ed"; Revised and updated by Tom Badgett and Todd Thomas, with Corey Sandler; ISBN-0-471-46912-2; John Wiley & Sons, Inc.; 2004.

9. Recursos estudiantiles

Para información sobre recursos estudiantiles disponibles en la UCR, incluyendo el Sistema de bibliotecas y la normativa universitaria vigente, favor visitar la página: https://www.ecci.ucr.ac.cr/vida-estudiantil/servicios-institucionales-para-estudiantes/guia-de-recursos-estudiantiles-de-la-ucr