Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI1201 - Programación II
I semestre 2013. Grupo 03: K,V 15:00-17:00. 303-IF.
Profesor Jeisson Hidalgo-Céspedes
Casillero: 04. Correo: jeissonh@gmail.com

Carta al estudiante

Objetivos

Introducir los fundamentos teóricos de programación para entrenar a cada estudiante en las técnicas básicas de construcción de programas, en especial en la especificación e implementación de módulos y artefactos de programación reutilizables, con el fin de que puedan usar herramientas de programación y participar en cualquier equipo dedicado a construir programas de mediana complejidad.

Al finalizar el curso el estudiante será capaz de:

1. Comprender los siguientes conceptos: memoria estática, memoria dinámica, enlace estático, enlace dinámico, flujos de entrada-salida, archivos de acceso aleatorio, recursividad, clases concretas, clases abstractas, clases parametrizadas o programación genérica, clases contenedoras e iteradores.
2. Elaborar especificaciones de clases completas, correctas, precisas y sin ambigüedades, utilizando una herramienta automatizada para generación de documentación.
3. Construir clases abstractas y clases que las derivan usando herencia y polimorfismo.
4. Usar clases parametrizadas pertenecientes a una biblioteca de uso difundido o estandarizado.
5. Construir programas basados en los algoritmos de inserción, eliminación y búsqueda sobre árboles binarios de búsqueda.
6. Usar excepciones al construir clases y programas para mejorar su robustez incorporando validación de entrada de datos y manejo de errores.
7. Aplicar casos de prueba para depurar clases concretas y abstractas.
8. Aprovechar las fortalezas de varios paradigmas de programación a distintos problemas.

Contenidos

1. Generalidades
    a. Descripción del uso de la memoria estática y la memoria dinámica en el lenguaje de programación que se usará en el curso.
    b. 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 que se usarán en el curso.
    d. Descripción del depurador disponible en al menos uno de los ambientes de programación que se usará en el curso.
    e. Descripción de los distintos mecanismos disponibles en el lenguaje de programación que se usará 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 que se usará en el curso.
2. Especificación
    a. Pautas para la especificación de clases concretas y clases abstractas.
    b. Discusión de las pautas para la especificación de clases parametrizadas.
3. Herencia y polimorfismo
    a. Definición de clases abstractas.
    b. Definición de clases que derivan de otras, ya sea abstractas o concretas.
    c. Definición de tipos polimórficos.
    d. Definición de métodos (y operadores, si el lenguaje lo permite) polimórficos.
4. Parametrización de clases
    a. Descripción general de una biblioteca de clases parametrizadas de uso estándarizado en el lenguaje de programación que se usará en el curso.

    b. 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.
5. Árboles
    a. Algoritmos de inserción, búsqueda, eliminación y el recorrido en orden para árboles binarios ordenados no balanceados.
6. Validación de entrada de datos y manejo de excepciones
    a. Defincón de clases de excepciones.
    b. Descripción del levantamiento de excepciones.
    c. Descripción de la captura y tratamiento de excepciones.
7. Manejo de archivos planos
    a. Por medio de flujos de datos.
    b. Por medio de acceso aleatorio.
8. 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.

Metodología de trabajo

El curso se compone de 16 semanas durante las cuales el docente presentará los conceptos y las técnicas de programación principales.

Al lo largo del ciclo lectivo el estudiante llevará a cabo proyectos de programación con el fin de poner en práctica y aplicar los conceptos y las técnicas del curso.

Bibliografía

1. Deitel, H.M.; Deitel, P.J. *C++ How to Program*, 8th edition. Prentice-Hall, 2012. http://www.deitel.com/.
2. Stroustrup, Bjarne. *The C++ Programming Language*, 3rd edition. Addison-Wesley; 1998. http://www.research.att.com/~bs/3rd.html.
3. Kernighan, Brian; Ritchie, Dennis. *El lenguaje de programación C*, 2da edición. Pearson, México, 1991.

Evaluación

- 15%. **Quices**. Ejercicios cortos que deberá el estudiante resolver en forma individual al iniciar la clase. No se repondrán por llegadas tardías.
- 20%. **Tareas cortas**. Ejercicios que deberá resolver el estudiante individualmente de forma extraclase.
- 25%. **Proyectos**. Dos proyectos de mediana complejidad. El profesor indicará si se pueden realizar en parejas o individualmente. El profesor también indicará si la ponderación será en proporción a su dificultad.
- 40%. **Exámenes parciales**. Tres exámenes de igual ponderación, cuyas fechas serán comunicadas oportunamente por el profesor.

Observaciones

1. En toda asignación, sea en papel o digital, se evaluará la indentación, uso correcto de paréntesis (redondos, cuadrados y llaves), la eficiencia, la elección de identificadores significativos, y las buenas prácticas de programación.
2. La nota `N` de una tarea o proyecto entregado `d` días tarde se calculará como $N = x - 10*d^2$, donde `x` es la nota que habría obtenido si se hubiere entregado a tiempo.
3. Cualquier asignación donde se detecte plagio, su calificación será anulada por completo (no sólo el o los ejercicios donde se detecte el plagio).