PROGRAMACIÓN Y ALGORITMOS I

CICLO	CLAVE DE LA ASIGNATURA
SEMESTRE 1	**C16PRG1**

OBJETIVO(S) GENERAL(ES) DE LA ASIGNATURA

El curso tiene como objetivo dar a los estudiantes un *background* sólido en programación, y en particular en programación en C y C++.

TEMAS Y SUBTEMAS

1. **Introducción**

   Estructura y sintaxis de un programa en C; Archivos de cabecera; Palabras reservadas; Tipos de datos; Variables, declaración; Alcance de las variables, locales y globales; Tipos de expresiones. *Lvalue, rvalue*; Operadores aritméticos, lógicos, relacionales y misceláneos; *printf* y *scanf*; Compilación; Redirección de la entrada y salida estándar; Errores comunes; Tiempo de compilación y tiempo de ejecución; *argv* y *argc*.

   Condiciones; *if*, *else*; *for*; *do while*; *break* y *continue*; *while*.

2. **Funciones**

   Declaración y definición de funciones; Partes de una función; Paso de argumentos por copia; Archivos cabecera propios; Compilación con varios archivos de declaración y definición de funciones.

3. **Memoria y arreglos**

   Heap, stack, code, globals; Notas sobre eficiencia; Arreglos; Direcciones de memoria; Arreglos en funciones; Arreglos en ciclos; Arreglos multidimensionales.

4. **Lectura y escritura de archivos**

   Apertura y modos de apertura; Lectura; Escritura; Archivos binarios.

5. **Apuntadores y memoria dinámica**

   Apuntadores; Operadores *, & y []; Apuntador *NULL*; Memoria dinámica; *malloc* y *free*; Arreglos con memoria dinámica; Aritmética de apuntadores; Apuntadores y archivos de texto.

6. **Cadenas de caracteres**

   Caracteres especiales y secuencias de escape; Manejo de strings; *string.h.*

7. **Estructuras, uniones**

   Tipos *enum*; *typedef struct*; memoria en la estructura; Funciones que reciben y devuelven estructuras; *typedef*; Estructuras y apuntadores; Aritmética de apuntadores en estructuras; *union*; estructuras y uniones; Arreglos de estructuras.

8. **Clases de almacenamiento**

   auto, *register*, *static*, *extern*.

9. **Apuntadores a funciones**

   Sintaxis; Funciones que reciben y devuelven apuntadores a funciones; *typedef*.

10. **Funciones con número variable de argumentos**

    va_list; *va_start*,*va_end.*

11. **Operaciones con bits**

    Operadores; Generadores de pseudo-aleatorios.

12. **Preprocesador**

    Macros; El preprocesador de C; directivas en gcc; Macros predefinidas; Concatenación; Definición condicional de macros; Compilación condicional.

13. **Algoritmos y complejidad**

    Invariante de ciclo; Orden; *Merge Sort* (recursión); *Insertion sort*.

14. **Estructuras de datos y recursividad**

    Listas ligadas; Listas doblemente ligadas; Algoritmos recursivos para (manejo de memoria en) listas ligadas; Algoritmo de Shunting-Yard; Árboles binarios.

15. **Introducción a la programación orientada a objetos**

    Conceptos OOP; Objeto; Abstracción; Encapsulamiento; Herencia; Polimorfismo; Sobrecarga de funciones y operadores; Compilación con g++; Tipos de datos; namespace.

16. **Clases**

class; Datos y funciones miembro; Constructor; Modificadores de acceso; Inicializaciones; Sobrecarga del constructor; Constructor de copia; Destructor; Funciones *friend*; Funciones *inline*; Apuntador this; Miembros *static*.

17. **Herencia**

    Clases base y heredada.

18. **Sobrecarga de funciones y operadores**

    Sobrecarga de funciones; Sobrecarga de operadores; Operadores sobrecargables.

19. **Polimorfismo, Abstracción, Encapsulamiento, Interfaces**

    Polimorfismo de funciones en clases derivadas; Funciones virtuales.

    Ejemplos de abstracción mediante etiquetas de acceso.

    Ejemplos de encapsulamiento.

    Clases abstractas; Funciones puramente virtuales.

20. **Temas misceláneos**

    Makefiles; Multithreading.

ACTIVIDADES DE APRENDIZAJE

Cursos presenciales

Desarrollo de proyectos

Resolución de ejercicios

Desarrollo de software

Preparación de presentaciones

Desarrollo de informes

Lectura de publicaciones especializadas

CRITERIOS Y PROCEDIMIENTOS DE EVALUACIÓN Y ACREDITACIÓN

40% Tareas.

30% Proyectos (10% el primero, 20% el final).

30% Exámenes.

BIBLIOGRAFÍA

R. Sedgewick, *Algorithms in C++*, Addison Wesley, 1998.

B. Preiss, *Data Structures and Algorithms with Object Oriented Design Patterns in C++*, Wiley, 1998. [http://www.brpreiss.com/]

C. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, 3rd edition, 2009.

J. Kleinberg and E.Tardos, *Algorithm Design*, Pearson, 2005.

D.Knuth, *The Art of Computer Programming, Vol.1 Fundamental Algorithms, Vol.3 Sorting and Searching*, Addison-Wesley, 3rd edition, 2011.

B. Kernighan and D. Ritchie, *The C programming language*, Prentice-Hall, 1988.

D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys* (CSUR) 23.1 (1991): 5-48.

U. Drepper. *What every programmer should know about memory*. Red Hat, Inc 11 (2007): 2007.

M. Weis, *Efficient C Programming: A practical approach*, Prentice Hall, 1994.

J. Pitt-Francis and J. Whiteley, *Guide to scientific computing in C++*, Springer Science & Business Media, 2012.