Estructura de Datos y Algoritmos 1

Teórico #1:

Introducción a los Algoritmos

Objetivos generales del curso

Presentar al estudiante y trabajar en:

- estructuras de datos y algoritmos esenciales para la resolución de problemas en computación, de manera eficiente;
- el rol de la abstracción de datos en el diseño y el desarrollo de sistemas.

Este curso profundiza conceptos de programación vistos en materias previas, tales como la programación recursiva e iterativa y sienta las bases para cursos posteriores.

Tópicos del curso (I)

- Introducción:
  - Abstracción en programación: particiones - refinamientos.
  - Abstracción procedural e introducción a la abstracción de datos.
- Análisis de Algoritmos
  - Eficiencia en espacio de almacenamiento y tiempo de ejecución. Orden de tiempo de ejecución del peor caso y caso promedio. Cálculo de tiempo de ejecución para algoritmos iterativos y recursivos
- Inducción y recursión
  - Recursión en un sistema computacional. Stack de ejecuciones. Análisis de iteración vs recursión.
  - Definición de tipos de datos inductivos.
  - Programación recursiva: Precondiciones, tipos, aplicaciones.

Tópicos del curso (I)

- Estructuras Dinámicas:
  - Estructuras estáticas y estructuras dinámicas. Punteros y manejo de memoria dinámica. Definición de estructuras lineales y no lineales (árboles). Algoritmos sobre dichas estructuras.
- Introducción a Tipos Abstractos de Datos (TADs)
  - Abstracción procedural y abstracción de datos.
  - Especificación de TADs: uso de pre y post condiciones. Implementación y uso de TADs. Ventajas de la programación con TADs.
- TADs Fundamentales
  - Especificación e implementación eficiente de TADs fundamentales: Listas, Pilas, Colas, Conjuntos, Diccionarios y Funciones Parciales. Variantes. Aplicaciones

Lenguaje de programación

- Se usará C++, sin incluir la parte del lenguaje referida al paradigma orientado a objetos propiamente
- Se usarán esencialmente las construcciones de C pero incorporando algunos elementos de C++, que se irán introduciendo a medida que se desarrollen los temas del curso.
- El objetivo principal es trabajar con abstracciones y estructuras de datos y algoritmos, y no enseñar un lenguaje.
- Entorno de desarrollo (IDE): Microsoft Visual Studio

Metodología

- 6 horas semanales de clases teórico/prácticas con resolución de ejercicios y trabajo en laboratorio
- Consultas semanales con ayudantes de cátedra y docentes del curso a través de medios de comunicación oficiales:
  - Foros de Aulas
  - Microsoft Teams
  - Correo electrónico (@fi365.ort.edu.uy)
  - No se permite el uso de Whatsapp no otros medios no oficiales.
- Evaluaciones
  - Exámenes Parciales.
  - Entregas obligatorias de ejercicios prácticos.

Ver y consultar periódicamente el sitio web oficial del curso en **Aulas** para novedades

Materiales bibliográficos

- Materiales en AULAS
  - Guías de clases teóricas, Lecturas, Videos, Ejercicios, Foros

- Libros básicos
  - **Cómo Programar en C/C++**, H.M Deitel & P.J. Deitel
  - **Data Structures and Algorithm Analysis in C and C++**, Mark Allen Weiss
  - **Estructuras de Datos y Algoritmos**, A. Aho, J.E. Hopcroft & J.D. Ullman

Entregas, Evaluaciones

Evaluación	Fecha entrega/evaluación	Puntos
Ejercicios Parte 1	Por AULAS hasta las 21:00 1. 1/Octubre (Entrega parcial opcional) 2. 15/Octubre (Entrega oficial)	10
Parcial 1	14/Octubre (Presencial) Mat. 14:00 hs. - Noct: 18:30 hs.	30
Ejercicios Parte 2	Por AULAS hasta las 21:00 1. 26/Noviembre (Entrega oficial)	15
Parcial 2	9/Diciembre (Presencial) Mat. 9:00 hs. - Noct: 18:30 hs.	45

Honestidad Académica

Las interacciones benefician el dominio de los temas del curso, pero hay una línea clara entre obtener ayuda de otros y aprovecharte de su trabajo.

Es razonable comunicarte con compañeros de clase, solicitar ayuda a los docentes, discutir el material para comprenderlo mejor, buscar materiales en la web e incluso colaborar con compañeros para discutir soluciones a los efectos de identificar problemas y fortalecer los resultados.

Sin embargo, la esencia de todo trabajo que entregues debe ser tuya, por lo que no está permitido acceder y utilizar una solución de otros, ni brindar una solución a otros.

Honestidad Académica

Esto incluye el uso de herramientas de IA, las cuales están permitidas y son promovidas como herramienta de apoyo al aprendizaje, pero no para la resolución de los trabajos individuales.

Recuerda verificar el contenido generado por la IA ya que no siempre es correcto o preciso. Además, debes conocer los riesgos y desafíos, como la creación de “alucinaciones”, los peligros para la privacidad, las cuestiones de propiedad intelectual, los sesgos inherentes y la producción de contenido falso.

Al inscribirte al curso asumes el compromiso, responsabilidad y respeto que implica para contigo, tus compañeros y los docentes del curso.

Video Intro a los Algoritmos

¿Qué es un algoritmo?

Procedimiento computacional que partiendo de un conjunto de datos de entrada (instancias de un problema) y mediante un conjunto de sentencias (órdenes), produce un conjunto de valores de salida. Debe además poseer las siguientes características:

- El número de sentencias del algoritmo debe ser **finito** y éste debe de terminar al cabo de un número finito de pasos u operaciones.
- Cada sentencia debe estar **definida** con precisión, rigurosidad y sin ambigüedades.
- Puede tener cero o más datos de **entrada**.
- Debe arrojar como resultado al menos un valor de **salida**
- Cada operación ha de realizarse de forma **exacta** y en tiempo finito.

Algoritmos, programas, lenguajes

Un **programa** es un algoritmo especificado en un determinado lenguaje de programación para ejecutarse en una computadora.

Un **lenguaje de programación** es un conjunto de reglas *sintácticas*, que permiten formar sentencias con un valor *semántico* asociado. Se utiliza para escribir algoritmos que serán ejecutados en computadoras.

No es posible que una computadora pueda trabajar sin programas.

Los algoritmos, además de resolver problemas, deben resolverlos en un **tiempo viable**, esperado, que en la práctica sea útil.

Algoritmos, programas, lenguajes

KURT MEHLHORN AND PETER SANDERS - Algorithms and Data Structures - The Basic Toolbox

Algorithms are at the heart of every nontrivial computer application. Therefore, every computer scientist and every professional programmer should know about the basic algorithmic toolbox: structures that allow efficient organization and retrieval of data, frequently used algorithms, and basic techniques for modeling, understanding, and solving algorithmic problems.

THOMAS H. CORMEN - Introductions to algorithms

Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing.

Ejemplo de algoritmo

```
int X(int a, int b){
  int cont = 0;
  while (a > b){
    cont++;
    a = a - b;
  }
  return cont;
}
```

Para poder identificar qué hace, se tuvo que hacer una ejecución paso a paso. Esto es porque no está del todo **legible**.

Ejemplo de algoritmo

```
int DivisionEntera(int dividendo, int divisor){
  int cociente = 0;
  while (dividendo > divisor){
    cociente++;
    dividendo = dividendo - divisor;
  }
  return cociente;
}
```

¿Funciona siempre el algoritmo?

Cuando se define un algoritmo, debemos definir sus **precondiciones** y **poscondiciones**

Ejemplo de algoritmo

```
//PRE: dividendo >= 0, divisor > 0
//POS: Retorna cociente de dividendo/divisor
int DivisionEntera(int dividendo, int divisor){
  int cociente = 0;
  while (dividendo > divisor){
    cociente++;
    dividendo = dividendo - divisor;
  }
  return cociente;
}
```

Programa C++ que usa el algoritmo anterior

```
int main(){
  int a = 15;
  int b = 3;
  int cociente = DivisionEntera(a, b);
  cout << cociente; //Imprime en pantalla el valor
}
```

Ejercicio 1

Dados dos enteros, numerador y denominador, que componen una fracción; diseñar un algoritmo que simplifique dicha fracción.

Ejemplo: 18 / 12 → 3 / 2

Ejercicio 1 - Solucion

```
//PRE: a > 0, b > 0
//POS: Retorna MCD de a y b
int MCD(int a, int b){
  int num1 = max(a,b);
  int num2 = min(a,b);
  int res;
  do {
    int res = num2;
    num2 = num1 % num2;
    num1 = res;
  } while (num2 != 0)
  return res;
}
```

```
//PRE: n >= 0, d > 0
//POS: Imprime fracción n/d
//     simplificada
void simplificar(int n, int d){
  int mcd = MCD(n,d);
  n = n / mcd;
  d = d / mcd;
  cout << "Numerador: " << n;
  cout << "Denominador: " << d;
}
```

Ejercicio 2

a) Desarrollar un algoritmo que dado un número N, determine si es primo.

b) Desarrollar un algoritmo que dado un número natural N, retorne el número primo más cercano que sea mayor al propio número.

Ejercicio 2 - Solución

```
bool esPrimo(int N){
  int i = 2;
  while ((N % i == 0) && (i < N))
    i++;
  return (i == N);
}

int primoMayorMasCercano(int N){
  int num = N+1;
  while (!esPrimo(num))
    num++;
  return num;
}
```