Estructura de Datos

INTRODUCCIÓN A LOS TADS

Conceptos para la resolución de problemas.

Especificación del problema - Análisis, identificando:

- Datos de entrada.
- Datos de salida.
- Procesos.

Técnicas algorítmicas para resolver un problema:

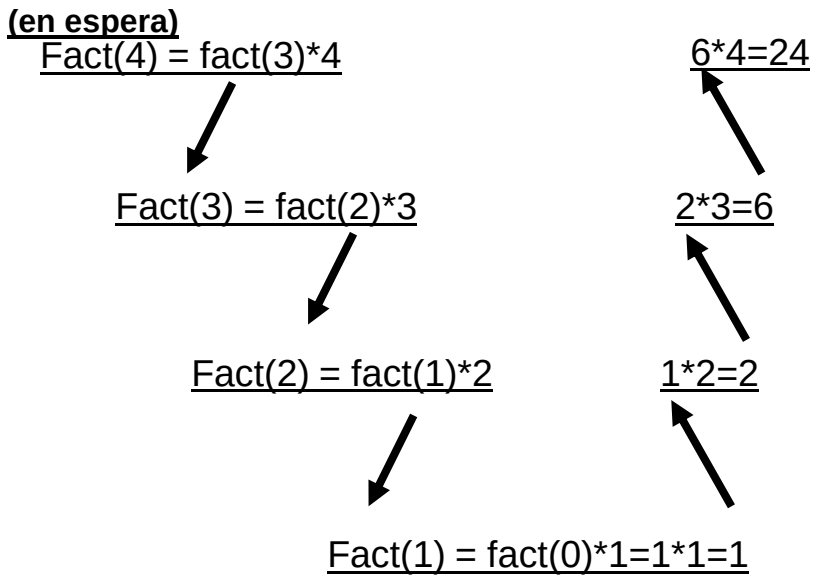
- Algoritmos de fuerza bruta → toman la ruta más evidente o corta para resolver el problema, independiente de que sea buena o no.
- Dividir y conquistar → la solución global es la unión de todas las soluciones parciales (subsoluciones).
- Programación dinámica → es similar al anterior, pero busca reutilizar los resultados obtenidos de los subproblemas.
- Algoritmos voraces → en cada punto de decisión, selecciona la opción que tiene el menos coste inmediato.
- Algoritmos probabilistas → además de los datos de entrada, dependen de valores producidos aleatoriamente. Cuando no es capaz de tomar una decisión óptima, escoge una al azar.

Conceptos para la resolución de problemas.

- **Recursión** → un algoritmo es recursivo si se encuentra definido en términos de sí mismo. Esta técnica se utiliza para las estructuras de datos recursivas, como los árboles.
  - Ventajas: simplicidad de expresión.
  - Desventajas: el espacio que utiliza en memoria por cada llamada de la rutina y tiempo adicional de ejecución.

Ejemplo:

```
Int factorial(int num)
{
   if (num==0)
        return 1;
   else
        return num*factorial(num-1);
}
```



Conceptos para la resolución de problemas.

Análisis de Algoritmos

→ Busca establecer la calidad de un programa y compararlo con otros que resuelvan el mismo problema, sin necesidad de desarrollarlos.

→ Permite evaluar el diseño de las estructuras de datos de un programa, midiendo su eficiencia.

→ Se basa en:

- **Las características estructurales del algoritmo que respalda al programa.**

  Se persigue encontrar una medida de calidad que permita comparar los algoritmos empleados, aplicando una función matemática para medir la eficiencia de cada algoritmo.

Conceptos para la resolución de problemas.

Eficiencia

- **Espacio de memoria que el algoritmo requiere,** se determina por:
  - Número y tamaño de las variables y estructuras de datos.

- **Tiempo de ejecución del algoritmo (tiempo de cómputo)** considerando:
  - La estructura del algoritmo, número de operaciones elementales que deben ser realizadas durante la ejecución del algoritmo.
  - La velocidad de operación del computador en que se ejecuta.
  - El compilador utilizado.
  - Tamaño de los datos de entrada con los que el programa trabaja.

    Se define como $T_A(n)$, tiempo empleado por el algoritmo A en procesar una entrada de tamaño n y producir una solución al problema.

- Tiempo y espacio suelen estar inversamente relacionados.

  ejemplo: para reducir los requisitos de espacio, se debe incrementar el tiempo de ejecución.

Conceptos para la resolución de problemas.

Técnica para medir el tiempo de ejecución:

- Complejidad Asintótica del tiempo de ejecución (complejidad en tiempo), busca medir el término de orden más alto que exprese el tiempo de ejecución, ya que con este se determina la tasa de crecimiento de la ecuación.
- Dependiendo de la complejidad en tiempo (eficiente v/s intratable) de un algoritmo, estos se clasifican en:
  - Algoritmos de tiempo polinómico, se mantienen estables.
  - Algoritmos de tiempo exponencial, crecen en tiempo de ejecución conforme al tamaño de la entrada.

Tipos Abstractos de Datos (TADs)

Definición:

→ *"Un TAD es un ente cerrado y autosuficiente, que no requiere de un contexto específico para que pueda ser utilizado en un programa, lo que garantiza portabilidad y reutilización del software y minimiza los efectos que puedan producir un cambio al interior del TAD"*

- → El ocultar la representación del tipo de datos respecto de las aplicaciones lo conocemos como **encapsulación de datos**.
- → Esta forma de trabajar tiene la ventaja de la **reutilización.**

Tipos Abstractos de Datos (TADs)

Definición:

→Es un modelo matemático de los objetos de datos que se necesitarán y las funciones que operan sobre estos objetos.

Ejemplo:

→El TAD conjunto y los elementos que lo constituyen (datos), posee las operaciones: UNION, INTERSECCION y DIFERENCIA DE CONJUNTOS.

→La implementación de un TAD se compone de:

→Las variables necesarias para definir los datos.

→Las rutinas de acceso a dichas variables.

Tipos Abstractos de Datos (TADs)

Observaciones:

- TAD: modelo matemático.
- Tipo de datos: implementación del modelo matemático anterior.
- Estructura de datos: colección de variables en memoria que están relacionadas de alguna forma específica.

Cada lenguaje de programación ofrece tipos de datos predefinidos.

Ejemplo para C:

- TAD: Entero → tipo de dato: Int

  Las operaciones que incluye son: suma, resta, multiplicación y división enteras, entre otras.

  Ahora, hay que considerar las limitaciones que tiene la implementación, como sólo puede ser el rango que posee el tipo de dato.

Tipos Abstractos de Datos (TADs)

Otros ejemplos:

- TAD: Vector → Implementación: Arreglos.
- TAD: Red → Implementación: Grafos.
- TAD: Matriz → Implementación: Arreglos bidimensionales.

Tipos Abstractos de Datos (TADs)

Clasificación de operaciones por grupos:

- Constructora: crea los elementos del TAD.
- Modificadora: permite alterar el estado de un elemento del TAD.
- Analizadora: permite consultar por el estado del objeto y retornar algún tipo de información.

→ Operaciones complementarias que permiten mayor portabilidad:

- Analizadoras:
  - Comparación de igualdad entre objetos.
  - Salida en pantalla, permite visualizar el estado de un elemento del TAD (sirve como base para alguna interfaz o depuración en pruebas).
- Modificadoras:
  - Copiar un objeto por otro, cambiando su estado.
  - Destrucción, retorna el espacio de memoria dinámica ocupada por un objeto abstracto. Después de su ejecución, el objeto deja de existir.
- Persistencia:
  - Operaciones que permiten guardar/leer el estado de un objeto abstracto desde un medio de almacenamiento secundario.

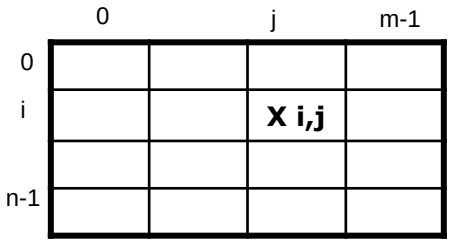
Tipos Abstractos de Datos (TADs)

Especificación genérica de un TAD:

- Nombre del TAD.
- Objeto abstracto.
- Restricciones.
- Lista de Operaciones.
- Definición de cada operación.

Ejemplo:

- TAD Matriz.
- Objeto abstracto:



- Restricciones: n>0, m>0
- Lista de Operaciones:
  - CrearMatriz(i, j)
  - AsignarMatriz(M, i, j, valor)
  - ObtenerDatoMatriz(M, i, j)
  - SumaMatriz(M1, M2)
  - MatrizNegativa(M)
  - RestarMatriz(M1, M2)
  - MatrizInversa(M)
  - MostrarMatriz(M)

Memoria

Memoria de almacenamiento secundario.

- Las estructuras de datos que se almacenan aquí se las llama estructuras de datos externas.

Memoria principal.

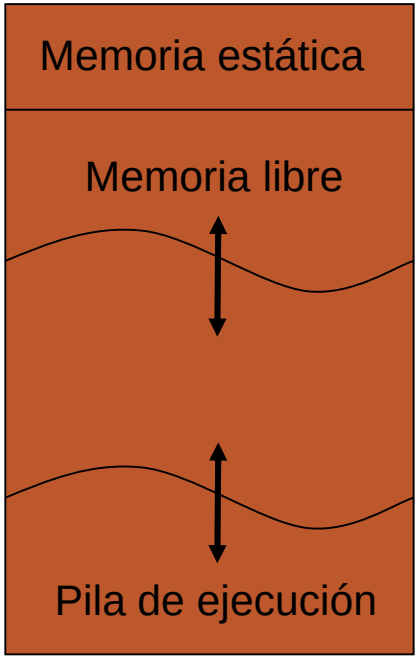
- Estructuras de datos internas.

Clasificación de Estructuras de Datos según tipo de memoria:

- Hablamos de una **Estructura de datos estática** cuando se le asigna una cantidad fija de memoria para esa estructura antes de la ejecución del programa. La cantidad de espacio asignado para la memoria estática se determina durante la compilación y no varía.
- Hablamos de una **Estructura de datos dinámica** cuando se le asigna memoria a medida que es necesitada, durante la ejecución del programa. En este caso la memoria no queda fija durante la compilación. El acto de asignar memoria durante la ejecución se llama **asignación dinámica de la memoria**.

Memoria

Memoria alta



Memoria baja

El sistema de la computadora mantiene una pila de ejecución, cuya cantidad de espacio que ocupa, variará durante la ejecución del programa.

- Cada vez que un proceso es invocado en un programa, se crea un registro de activación y se almacena en la pila de ejecución.
- El registro contiene espacio para todas las variables declaradas en un procedimiento y una copia a los parámetros que están siendo pasados a un procedimiento. Además de la indicación para continuar donde debe seguir la ejecución del programa, una vez se complete la función.

La memoria dinámica se almacena en la memoria libre, la que crece hacia la zona baja.

Se producen errores cuando:

- Demasiada memoria es asignada dinámicamente.
- Se crean demasiados registros de activación.