



**TECNOLÓGICO
NACIONAL DE MÉXICO**
CAMPUS MONCLOVA

Estructura de datos

Cristopher Zinedine Alvarez Sanchez

Diferentes estructuras de datos y su clasificación para elaborar un cuadro sinóptico.

¿Cuáles son los tipos de estructuras de datos?

Las estructuras de datos estáticas son aquellas en las que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa, mientras que una estructura de datos dinámica es aquella en la que el tamaño ocupado en memoria puede modificarse

Arreglos

Un array es un tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo y ordenado, es decir, todos ellos del mismo tipo y relacionados. Su condición de homogéneo, indica que sus elementos están compuestos por el mismo tipo de dato, y su condición de ordenado hace que se pueda identificar del primer al último elemento que lo compone.

Estructura de datos dinámicas

Por otro lado, vimos que en programación existen estructuras de datos dinámicas, es decir, una colección de elementos -nodos- que normalmente se utilizan para dejar asentados registros. A diferencia de un array que contiene espacio para almacenar un número fijo de elementos, una estructura dinámica de datos se amplía y contrae durante la ejecución del programa. Veamos algunos casos:

Estructura de datos lineales

Las estructuras de datos lineales son aquellas en las que los elementos ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y predecesor, es decir, sus elementos están ubicados uno al lado del otro relacionados en forma lineal.

Hay tres tipos de estructuras de datos lineales:

Listas enlazadas

Pilas

Colas

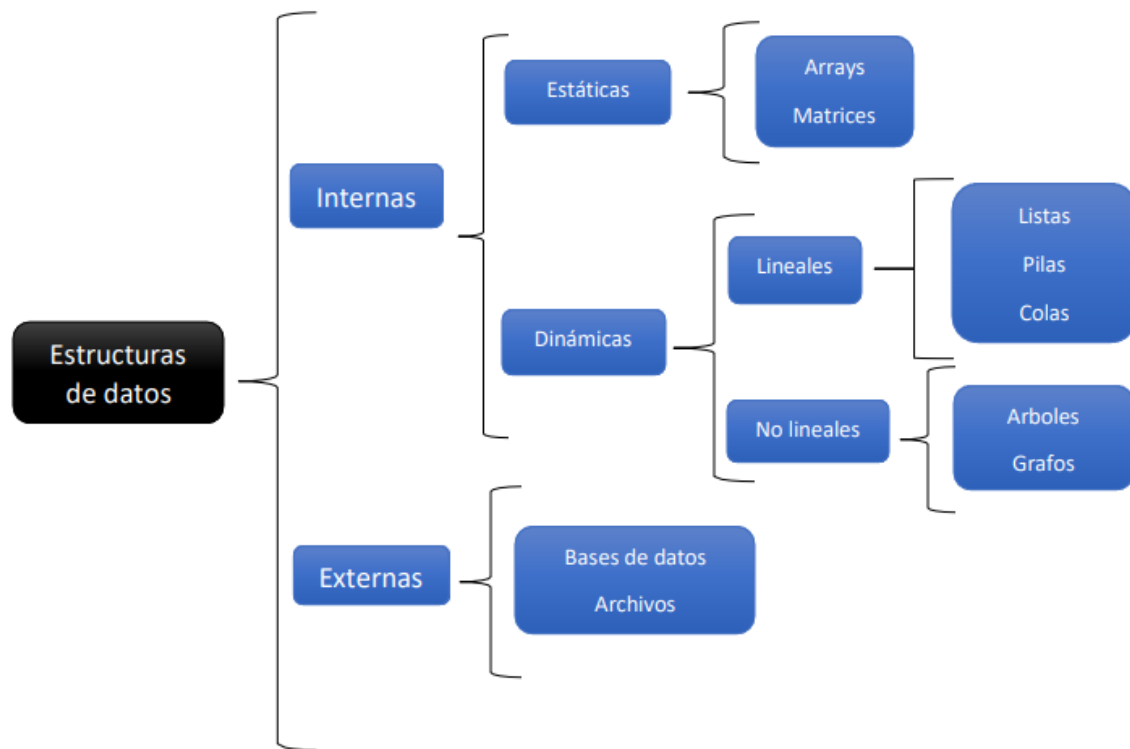
Estructura de datos no lineales

Las estructuras de datos no lineales, también llamadas multienlazadas, son aquellas en las que cada elemento puede estar enlazado a cualquier otro componente. Es decir, cada elemento puede tener varios sucesores o varios predecesores.

Existen dos tipos:

Árboles

Grafos



Resumen sobre el concepto de tipos de datos abstractos y su implementación en la programación orientada a objetos.

Listas

Una lista es una secuencia de nodos o elementos del mismo tipo, tal que, cada nodo señala, apunta, conoce, o sabe cuál es el siguiente nodo, (si existe), si no hay un nodo siguiente, entonces apunta a NULL, (una dirección segura).

Pilas

Una pila se define formalmente como una colección de datos a los cuales se puede acceder mediante un extremo, que se conoce generalmente como tope. Las pilas no son estructuras fundamentales de datos Para su representación requieren el uso de otras estructuras de datos, como arreglos o listas.

Colas

Una cola es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los dos extremos de la lista. Un elemento se inserta en la cola (parte final) de la lista y se suprime o elimina por la frente (parte inicial, cabeza) de la lista.

Arboles

Un árbol consta de un número finito de elementos, a estos se les llama nodos estos nodos están unidos por líneas llamadas ramas. Definición 1: Un árbol consta de un conjunto infinito de elementos llamados nodos, y un conjunto de líneas dirigidas llamadas ramas que conectan los nodos

Grafos

Un árbol consta de un número finito de elementos, a estos se les llama nodos estos nodos están unidos por líneas llamadas ramas. Definición 1: Un árbol consta de un conjunto infinito de elementos llamados nodos, y un conjunto de líneas dirigidas llamadas ramas que conectan los nodos

Diferencias entre el manejo de memoria estática y dinámica, así como ventajas y desventajas.

Memoria estática.

- Las técnicas de asignación de memoria estática son sencillas.

La asignación de memoria puede hacerse en tiempo de compilación y los objetos están vigentes desde que comienza la ejecución del programa hasta que termina.

En los lenguajes que permiten la existencia de subprogramas, y siempre que todos los objetos de estos subprogramas puedan almacenarse estáticamente se aloja en la memoria estática un registro de activación correspondiente a cada uno de los subprogramas.

Estos registros de activación contendrán las variables locales, parámetros formales y valor devuelto por la función.

Consideraciones

- ✓ Error en tiempo de ejecución de índice fuera del rango.
- ✓ Se debe conocer con anticipación el tamaño de la estructura.
- ✓ Se guardan en memorias adyacentes.
- ✓ Vectores, matrices, cubos, registros, archivos.

Ventajas

- La velocidad de acceso es alta.

- Para retener los datos solo necesita estar energizada.
- Lógica simple. Son más fáciles de diseñar.

Desventajas:

- ✓ No se puede modificar el tamaño de la estructura en tiempo de ejecución.
- ✓ No es óptimo con grandes cantidades de datos.
- ✓ Desperdicio de memoria cuando no se utiliza en su totalidad del tamaño $v[100]$.
- ✓ Menor capacidad, debido a que cada celda de almacenamiento requiere más transistores.
- ✓ Mayor costo por bit.
- ✓ Mayor consumo de Potencia

MEMORIA DINÁMICA

La memoria dinámica es un espacio de almacenamiento que se solicita en tiempo de ejecución. De esa manera, a medida que el proceso va necesitando espacio para más pág. 6 líneas, va solicitando más memoria al sistema operativo para guardarlas. El medio para manejar la memoria que otorga el sistema operativo, es el puntero, puesto que no podemos saber en tiempo de compilación dónde nos dará huecos el sistema operativo (en la memoria de nuestro PC).

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definirla cantidad de memoria a utilizar en el momento de compilar el programa.

Cuando se crea un programa en el que es necesario manejar memoria dinámica el sistema operativo divide el programa en cuatro partes que son: texto, datos (estáticos), pila y una zona libre o heap. En el momento de la ejecución habrá tanto partes libres como partes asignadas al proceso por lo cual si no se liberan las partes utilizadas de la memoria y que han quedado inservibles es posible que se “agote” esta parte y por lo tanto la fuente de la memoria dinámica. También la pila cambia su tamaño dinámicamente, pero esto no depende del programador sino del sistema operativo.

VENTAJAS:

- ✓ Es posible disponer de un espacio de memoria arbitrario que dependa de información dinámica (disponible sólo en ejecución): Toda esa memoria que maneja es implementada por el programador cuando fuese necesario.

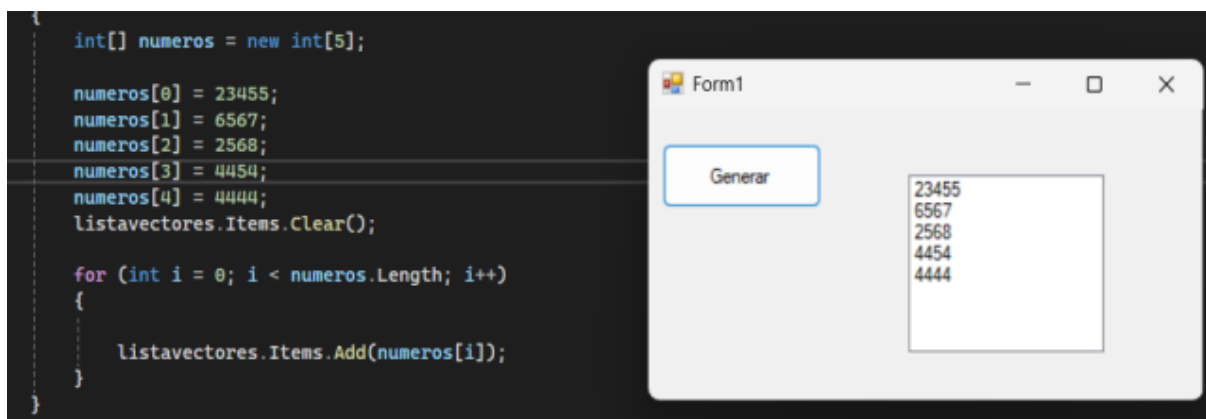
- ✓ Otra ventaja de la memoria dinámica es que se puede ir incrementando durante la ejecución del programa. Esto permite, por ejemplo, trabajar con arreglos dinámicos.
- ✓ Es memoria que se reserva en tiempo de ejecución. Su tamaño puede variar durante la ejecución del programa y puede ser liberado mediante la función free.

DESVENTAJAS:

- ✓ Es difícil de implementar en el desarrollo de un programa o aplicación.
- ✓ Es difícil implementar estructuras de datos complejas como son los tipos recursivos (árboles, grafos, etc.). Por ello necesitamos una forma para solicitar y liberar memoria para nuevas variables que puedan ser necesarias durante la ejecución de nuestros programas: Heap.
- ✓ Una desventaja de la memoria dinámica es que es más difícil de manejar.
- ✓ La memoria dinámica puede afectar el rendimiento. Puesto que con la memoria estática el tamaño de las variables se conoce en tiempo de compilación, esta información está incluida en el código objeto generado. Cuando se reserva memoria de manera dinámica,
- ✓ Se tienen que llevar a cabo varias tareas, como buscar un bloque de memoria libre y almacenar la posición y tamaño de la memoria asignada, de manera que pueda ser liberada más adelante. Todo esto representa una carga adicional, aunque esto depende

Programa para la generación de vectores en tiempo de ejecución.

Este sencillo código muestra la generación de vectores en tiempo de ejecución esta hecho en c# Windows forms y usa un botón junto con un listbox que muestra los vectores.



Programa que ejemplifique el uso de la memoria dinámica.

Este sencillo código en c# consola muestra de una manera sencilla la manera de usar la memoria dinámica de cola.

```

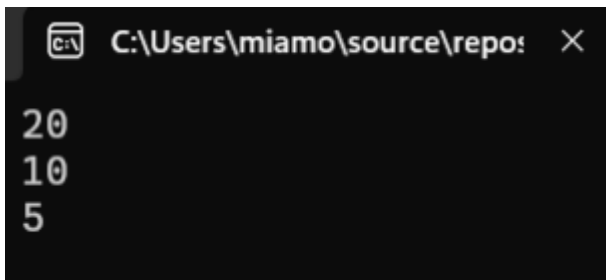
using System;
using System.Collections;

namespace EjemploCola
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            Queue cola = new Queue();
            cola.Enqueue(20);
            cola.Enqueue(10);
            cola.Enqueue(5);

            while (cola.Count > 0)
            {
                int dato = (int)cola.Dequeue();
                Console.WriteLine(dato);
            }
            Console.ReadKey();
        }
    }
}

```

Aquí sale en orden de la cola



```

C:\Users\miamo\source\repo: X
20
10
5

```

Identificar la eficiencia de dos algoritmos distintos que resuelven el mismo problema en un cuadro comparativo.

Búsqueda lineal: Recorre los elementos uno por uno hasta encontrar el elemento deseado

Búsqueda binaria: Divide repetidamente la lista en mitades y compara el valor buscado con el valor medio.

Ambos te pueden ayudar a buscar elementos en una lista ordenada o desordenada de una manera eficiente.

Característica	Búsqueda lineal	Búsqueda binaria
Tipo de estructura	Funciona en listas ordenadas o no ordenadas	Requiere una lista ordenada
Eficiencia promedio	$O(n)$ – lineal (peor caso y caso promedio)	$O(\log n)$ logarítmica (peor caso y caso promedio)
Ventajas	Simple de implementar	Muy eficiente en listas grandes y ordenadas
Requisito previo	Ninguno	La lista debe de estar ordenada previamente
Desventajas	Ineficiente en listas grandes y ordenadas	Requiere un ordenamiento previo, no funciona en listas no ordenadas
Uso común	Listas pequeñas y no ordenadas	Listas grandes y ordenadas

Bibliografía.

<https://blog.soyhenry.com/que-es-una-estructura-de-datos-enprogramacion/#:~:text=Las%20estructuras%20de%20datos%20est%C3%A1ticas,o cupado%20en%20memoria%20puede%20modificarse>

<http://adrian-estructuradedatos.blogspot.com/2011/04/memoria-estatica-y-dinamica.html>