INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES

DE MONTERREY

TC1031 - Programación de estructuras de datos y algoritmos fundamentales

Profesor: David Sánchez

Actividad 2 - Implementación de una tabla Hash

Grupo 604

Diana María Arámburo Lozano | A01646337

Camila Gomez Godinez | A01639319

Gabriela Ruelas Gaytán | A01640880

Emilio Guzmán Flores | A01643485

Samantha Mailen Gallardo Mota | A01640886

Sábado 13 de septiembre del 2025

Actividad 2 - Implementación de una tabla Hash

Explicación breve de la hash tabla

La hash table almacena pares (key, value) en un arreglo. Para ubicar cada clave, aplica una función hash que transforma la clave en un índice del arreglo.
La idea central es acceder por índice del arreglo es O(1), así que, si la función hash dispersa bien y controlamos las colisiones, las operaciones CRUD (crear, leer, actualizar, eliminar) son muy rápidas.
Las colisiones suceden cuando dos claves mapean al mismo índice. Cómo usamos open addressing, si la casilla está ocupada, buscamos otra casilla siguiendo una ruta de sondeo (probing) hasta hallar un lugar libre o la clave objetivo.
Estados de casilla:

- EMPTY (nunca usada): la búsqueda puede detenerse.
- OCCUPIED (ocupada): hay par clave–valor.
- DELETED (tombstone): hubo un elemento; se conserva para no romper futuras búsquedas y puede reutilizarse al insertar.

Justificación del método de colisiones elegido (Quadratic Probing)

Elegimos closed hashing (open addressing) con quadratic probing:

- **Simplicidad:** solo se usa un arreglo, no hay listas enlazadas ni estructuras auxiliares.
- **Menos clustering primario:** respecto al sondeo lineal, el desplazamiento cuadrático espacia mejor las pruebas y reduce "tiras" largas de ocupación contigua.
- **Buen rendimiento práctico:** con un factor de carga moderado, las longitudes de sondeo suelen ser cortas, manteniendo tiempos cercanos a O(1).
- **Eliminación segura:** con tombstones (DELETED) no se rompen las rutas de búsqueda de otras claves; a la vez, esos huecos pueden reaprovecharse en inserciones posteriores.

Complejidad de insert, get y remove

Sea *m* la capacidad de la tabla y *n* el número de elementos, con factor de carga $\alpha = n/m$:

insert(key, value)

Promedio: O(1) amortizado (con α moderado).
Peor caso: O(m) si la tabla está casi llena o hay mala dispersión.

get(key)

Promedio: O(1).
Peor caso: O(m) (muchas colisiones o larga cadena de tombstones).

remove(key)

Promedio: O(1).
Peor caso: O(m) (misma razón: recorrido largo hasta hallar/descartar).