Facultad:	Ingeniería
Escuela:	Computación
Asignatura:	Programación con Estructuras de Datos

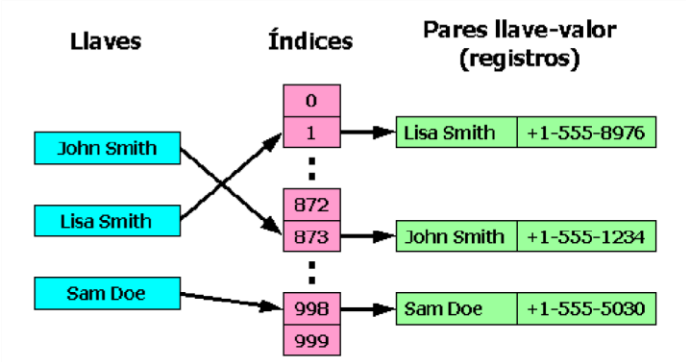
Tema: **“Tablas Hash”**

Competencia

Desarrolla sistemas de información informáticos mediante la integración de principios matemáticos, ciencia computacional y prácticas de ingeniería, considerando estándares de calidad y mejores prácticas validadas por la industria del software.

Introducción Teórica

Una tabla hash o mapa hash es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada usando el nombre, número de cuenta o id. Funciona transformando la clave con una función hash en un hash, un número que la tabla hash utiliza para localizar el valor deseado.



Las tablas hash se suelen implementar sobre arreglos de una dimensión, aunque se pueden hacer implementaciones multi-dimensionales basadas en varias claves. Como en el caso de los

arrays, las tablas hash proveen tiempo constante de búsqueda promedio O(1) (esto significa alta eficiencia y respuesta rápida) sin importar el número de elementos en la tabla.
Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información.

Las tablas hash almacenan la información en posiciones pseudo-aleatorias, así que el acceso ordenado a su contenido es bastante lento.

Funcionamiento:

Las operaciones básicas implementadas en las tablas hash son:

```
inserción(llave, valor)
búsqueda(llave) que devuelve valor
```

La mayoría de las implementaciones también incluyen `borrar(llave)`. También se pueden ofrecer funciones como iteración en la tabla, crecimiento y vaciado. Algunas tablas hash permiten almacenar múltiples valores bajo la misma clave.

Para usar una tabla hash se necesita:

- ✓ Una estructura de acceso directo (normalmente un array).
- ✓ Una estructura de datos con una clave.
- ✓ Una función resumen (hash) cuyo dominio sea el espacio de clases y su imagen (o rango) los números naturales.

Inserción:

1. Para almacenar un elemento en la tabla hash se ha de convertir su clave a un número. Esto se consigue aplicando la **función resumen** a la clave del elemento.

2. El resultado de la función resumen ha de *mapearse* al espacio de direcciones del **arreglo** que se emplea como soporte, lo cual se consigue con la función módulo. Tras este paso se obtiene un índice válido para la tabla.
3. El elemento se almacena en la posición de la tabla, obtenido en el paso anterior.
    a. Si en la posición de la tabla ya había otro elemento, se ha producido una colisión. Este problema se puede solucionar asociando una **lista** a cada posición de la tabla, aplicando otra función o buscando el siguiente elemento libre. Estas posibilidades han de considerarse a la hora de recuperar los datos.

Búsqueda:

1. Para recuperar los datos, es necesario únicamente conocer la clave del elemento, a la cual se le aplica la **función resumen**.
2. El valor obtenido se mapea al espacio de direcciones de la tabla.
3. Si el elemento existente en la posición indicada en el paso anterior tiene la misma clave que la empleada, entonces es el deseado. Si la clave es distinta, se ha de buscar el elemento según la técnica empleada para resolver el problema de las colisiones al almacenar el elemento.

Para utilizar las tablas hash en C# necesitamos el nombre de espacio Collections; El cual lo invocamos de la siguiente forma

```
using System.Collections;
```

Sintaxis de métodos que utiliza las tablas hash en C#

- Para crear la tabla hash:
    - ***Hashtable nombretable = new Hashtable()***
- Para agregar valores.
    - ***Nombretable.Add(clave, valor);***

- Para accedes a los valores por medio de la clave.
    - ***Nombretable [clave]***
- Quitar un par llave/valor de la colección.
    - ***Nombretable.Remove(clave)***
- Saber si la colección contiene un par cuya **clave** sea la que se le pasa como parámetro.
    - ***Nombretable.ContainsKey(clave)***
- Contiene un par cuyo valor sea el que se le pasa como parámetro.
    - ***Nombretable.ContainsValue (valor)***

Materiales y Equipo

- Guía de Laboratorio Nº 8
- Computadora con programa: Visual Studio C#
- Dispositivo de Almacenamiento (USB).

Procedimiento

Ejemplo 1:

1. Abrir un proyecto en modo consola en Visual Studio C#
2. Agregar la libería `using System.Collections;`
3. Dentro del main digitar lo siguiente

```
// Crear una tabla hash utilizando la clase Hashtable disponible en Collections

Hashtable thash1 = new Hashtable();/*La tabla se llama thash1, pero puede ser cualquier nombre*/

// Agregando elementos a la tabla hash
//los valores pueden duplicarse pero no las claves
thash1.Add("txt", "Programa notepad.exe");
thash1.Add("bmp", "Programa paint.exe");
thash1.Add("dib", "Programa paint.exe");
thash1.Add("rtf", "Programa wordpad.exe");
```

```
//Comprobación de validación de que no se admiten llaves duplicadas

try
{
    thash1.Add("txt", "Programa winword.exe");
}
catch
{
    Console.WriteLine("Un elemento con la clave = \"txt\" Ya existe \n");
}

//podemos obtener el valor utilizando la clave
Console.WriteLine("-----------------------------------------------");
Console.WriteLine("Imprimimos un valor de la tabla usando la clave");
Console.WriteLine("Para la Clave = \"rtf\", valor = {0} ", thash1["rtf"] +"\n");

// También podemos cambiar el valor utilizando la clave
thash1["rtf"] = "recortes.exe";
Console.WriteLine("-----------------------------------------------");
Console.WriteLine("Cambiando valor asociado a clave rtf");
Console.WriteLine("Para la clave= \"rtf\", valor = {0}.", thash1["rtf"]);
Console.WriteLine("\n-----------------------------------------------");

//si la clave no existe, al utilizar esta sintaxis también se puede agregar igual que add
thash1["doc"] = "winword.exe";

// ContainsKey puede ser usada para probar
// si una clave ya existe

if (!thash1.ContainsKey("ht"))
{
    thash1.Add("ht", "hypertrm.exe");
    Console.WriteLine("Valor Agregado para =\"ht\": {0}", thash1["ht"]);
    Console.WriteLine("-----------------------------------------------");
}

// para obtener solo valores, utilice la propiedad values
//utilizamos ICollection para formar una nueva coleccion
//a partir de los valores de la tabla hash
ICollection valueColl = thash1.Values;
```

```
//imprimiendo solo valores

Console.WriteLine();
Console.WriteLine("-----------------------------------------------");
Console.WriteLine("IMPRIMIENDO SÓLO VALORES DE LA TABLA");
foreach (string s in valueColl)
{
    Console.WriteLine("Value = {0}", s);
}

// para obtener solo las claves utilice la propiedad Keys
//utilizamos ICollection para formar una nueva coleccion
//a partir de las claves de la tabla hash
ICollection keyColl = thash1.Keys;
Console.WriteLine();
Console.WriteLine("-----------------------------------------------");
Console.WriteLine("IMPRIMIENDO SÓLO LAS LLAVES DE LA TABLA");
foreach (string s in keyColl)
{
    Console.WriteLine("Clave = {0}", s);
}

// Removiendo elementos
Console.WriteLine("-----------------------------------------------");
Console.WriteLine("REMOVER CLAVES");
Console.WriteLine("Primero se remueve la clave .doc");
Console.WriteLine("\nRemover (\"doc\")"); thash1.Remove("doc");
Console.WriteLine("-------");
Console.WriteLine("\n Después verificamos si la clave aún existe en la tabla");
if (!thash1.ContainsKey("doc"))
{
    Console.WriteLine("Clave \"doc\" no encontrada.");
}
Console.WriteLine("-----------------------------------------------");
Console.ReadLine();
```

Análisis de Resultados

Ejercicio 1:

Cree un nuevo proyecto en C# modo gráfico que simule un diccionario, el programa deberá contener el siguiente menú

1. Agregar Definición
2. Buscar Definición
3. Eliminar Definición
4. Mostrar definiciones existentes

Explicación menú:

1. Agregar Definición

Se debe agregar una palabra y su significado utilizando tablas hash, utilice la palabra a definir como clave y el significado como el valor.

Ejemplo

Jabón: es un producto que sirve para la higiene personal y para lavar determinados objetos. Jabón tendría que ser la clave y el significado seria el valor.

Tomar en cuenta que debe validar si una definición ya existe, en caso de existir enviar un mensaje.

2. Buscar Definición

Se deberá pedir una palabra a buscar y al encontrarla deberá imprimir su significado, en caso de no existir, deberá enviar un mensaje, La definición X no fue encontrada

3. Eliminar Definición

Deberá pedir una definición a eliminar, en caso de no existir dicha definición, enviar un mensaje, la definición X no existe

Investigación Complementaria

Crear un proyecto en Visual C# que contenga el siguiente menú

1. Agregar Datos
2. Actualizar Datos
3. Eliminar Datos
4. Mostrar Todos los datos
5. Consultar Datos

Los datos a utilizar serán DUI (clave) y nombre completo, al momento de actualizar los datos se deberá pedir el DUI para realizar la búsqueda y se modificara el nombre, se deben tener en cuenta todas las validaciones posibles