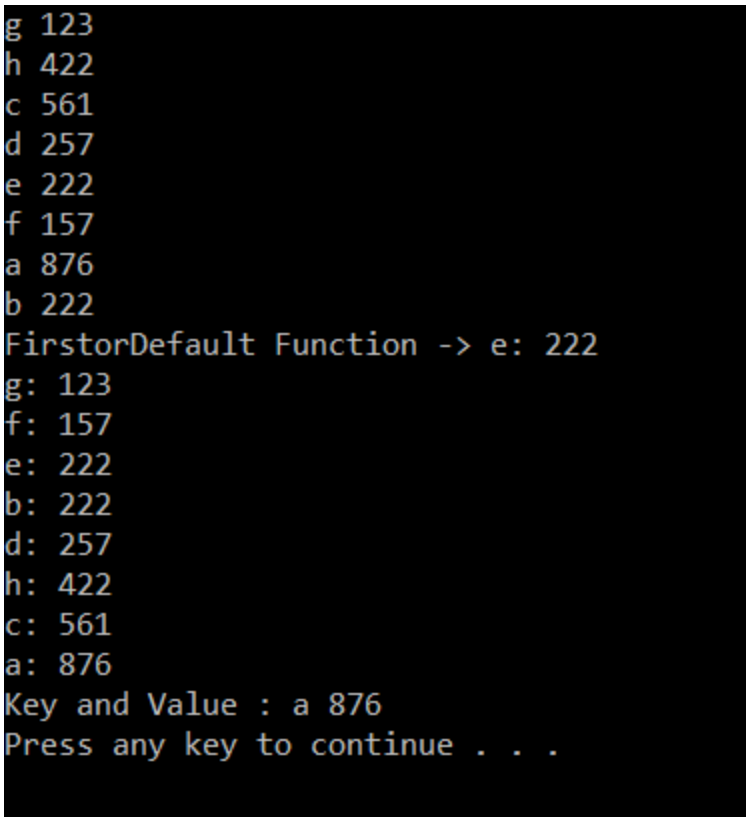
Dictionary

Dictionary

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TestDB
{
    public class class1
    {
     // attribute
        public int val { get; set; }
        public string key { get; set; }

     // method
        public void call(string _key)
        {
     // create dictionary (duplicate value 222)
        Dictionary<string, int> dic=new Dictionary<string, int>();
            dic.Add("g", 123);
            dic.Add("h", 422);
            dic.Add("c", 561);
            dic.Add("d", 257);
            dic.Add("e", 222);
            dic.Add("f", 157);
            dic.Add("a", 876);
            dic.Add("b", 222);

     // loop to see result
            foreach (KeyValuePair<string, int> pair in dic)
            {
                key = pair.Key;
                val = pair.Value;
                Console.WriteLine(key + " " + val);
            }

            dic.Keys.ToList();
            if (dic.ContainsKey(_key))
            {
                key = _key;
                val = dic[_key];
```

```
            }


      // get the first or default record of value 222)
      var myKey = dic.FirstOrDefault(x => x.Value ==222).Key;
            Console.WriteLine("FirstorDefault Function -> {0}: {1}",
myKey.ToString(), dic[myKey]);


      // ** Sorting **

            // Use LINQ to specify sorting by value. (Method #2
sorting)
            var items = from pair in dic
                                orderby pair.Value ascending
                                 select pair;

            // Display results.
            foreach (KeyValuePair<string, int> pair in items)
            {
                Console.WriteLine("{0}: {1}", pair.Key, pair.Value);
            }


            //// Use List to sort Key (Method #1 sorting)
            //var list = dic.Keys.ToList();
            //list.Sort();
            //// Loop through keys.
            //foreach (var key in list)
            //{
            //    Console.WriteLine("{0}: {1}", key, dic[key]);
            //}

        }
    };



     // main class
class Program
    {
        static void Main(string[] args)
        {
```

```
            class1 c = new class1();
            c.call("a");
            Console.WriteLine("Key and Value : " +c.key + " " +
c.val);
        }
    }
}
```



Two Dictionary

2 Dimension Dictionary

```
Dictionary<int, int> inner = new Dictionary<int, int>();
inner.Add(0, 1); // add item, if the key exist, it will throw error
inner[1] = 8; // this is also add new item or replace if exist

Dictionary<int, Dictionary<int, int>> dic = new Dictionary<int, Dictionary<int, int>>();
dic.Add(0,inner);
//dic[1].Add(5, 1); // throw error, keynotfound
//dic[1][0] = 1; // throw error, keynotfound
dic.Add(1, new Dictionary<int, int>() { { 0, 5 }, { 1, 4} });

dic[0].Add(4, 5); // since got key 0 in dic, directly can add in another row of
dictionary with key 2
dic[0][0] = 10; // reset the value of dictionary from 1 to 10


//---------------------- safety check if containskey of 0 ------------
if (dic.ContainsKey(0))
{
    if (dic[0].ContainsKey(0))
           dic[0][0] += 1; // add one number
    else
           dic[0].Add(0, 1);
}
else
{
    dic.Add(0, inner);
}

Console.WriteLine("Print dic[0]: ");
foreach (KeyValuePair<int, int> pair in dic[0])
{
    Console.Write("Key: " + pair.Key.ToString()+" | ");
    Console.WriteLine("Value: " + pair.Value.ToString());
}

Console.WriteLine("Print dic[1]: ");
foreach (KeyValuePair<int, int> pair in dic[1])
{
    Console.Write("Key: "+pair.Key.ToString() + " | ");
    Console.WriteLine("Value: " + pair.Value.ToString());
}

Console.WriteLine("Show All: ");
foreach(KeyValuePair<int, Dictionary<int, int>> pair in dic)
{
    foreach(KeyValuePair<int, int> pair_inner in pair.Value)
    {
         Console.Write("MainKey: " + pair.Key.ToString() + " | ");
         Console.Write("Key: " + pair_inner.Key.ToString() + " | ");
         Console.WriteLine("Value: " + pair_inner.Value.ToString());
     }
}
```

```
Console.ReadKey();
```

```
Print dic[0]:
Key: 0 | Value: 11
Key: 1 | Value: 8
Key: 4 | Value: 5
Print dic[1]:
Key: 0 | Value: 5
Key: 1 | Value: 4
Show All:
MainKey: 0 | Key: 0 | Value: 11
MainKey: 0 | Key: 1 | Value: 8
MainKey: 0 | Key: 4 | Value: 5
MainKey: 1 | Key: 0 | Value: 5
MainKey: 1 | Key: 1 | Value: 4
```

Dictionary vs Hashtable

Dictionary	Hashtable
Generic Type Dictionary<TKey, TValue> TKey: Type of keys in Dictionary TValue: Type of Values in Dictionary	Not Generic Type
Only public static members are thread safe	All members in Hashtable are thread safe
Return **error** if try to find a key which does not exist	Return **null** if the key is found does not exist
Faster than hashtable because no boxing and unboxing	Slower than dictionary because it requires boxing and unboxing

Dictionary

```
class Program
    {
        static void Main(string[] args)
        {
            Dictionary<string, int> d = new Dictionary<string, int>()
     {
           {"csharpstar", 2},  {"easywcf", 1}};
            // Loop over pairs with foreach
            foreach (KeyValuePair<string, int> pair in d)
            {
                Console.WriteLine("{0}, {1}", pair.Key, pair.Value);
            }
            foreach (var pair in d)
            {
                Console.WriteLine("{0}, {1}", pair.Key, pair.Value);
            }
            Console.ReadKey();
        }
    }
```

Hashtable

```
class Program
    {

        static Hashtable GetHashtable()
        {
            // Create and return new Hashtable.
            Hashtable hashtable = new Hashtable();
            hashtable.Add("csharpstar", 1);
            hashtable.Add("easywcf", 2);
            return hashtable;
        }
        public static void Main()
        {
            Hashtable hashtable = GetHashtable();
            // See if the Hashtable contains this key.
            Console.WriteLine(hashtable.ContainsKey("easywcf"));
            // Test the Contains method. It works the same way.
            Console.WriteLine(hashtable.Contains("csharpstar"));
            // Get value of csharpstar with indexer.
            int value = (int)hashtable["csharpstar"];
            // Write the value of Area.
            Console.WriteLine(value);
        }
    }
```