Dictionary<TKey,TValue>

Associative Collection

Topics

- Dictionary<Tkey,Tvalue>
- Creating Dictionaries
- Adding Items
- Looping
- Removing
- Checking for Keys

Dictionary<TKey, TValue>

C# Dictionary

- **Dictionary<TKey, TValue>**
  an **associative collection**.

- The *key* is ***associated*** with the *value*.

What are TKey and TValue?

They are generic type parameters

This Photo by Unknown Author is licensed under CC BY-NC-ND

C# Dictionary

When do you use a Dictionary?

When you need to look up data
using information about the data.

EX: using your social security # to look up your tax records

C# Dictionary

EXAMPLE:

We want to look up a specific student's information at Full Sail.

If we use an array to store student info, how would we find the info for a specific student? We would have to loop over the entire array. That performance is O(N) which is linear.

We need a faster way to look up data. The Dictionary would allow us to look up a specific student using the student's ID as the key. That performance is O(1) which is constant.

For More Info

- Generics
  https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/index

- Hash Tables
  https://en.wikipedia.org/wiki/Hash_table

Creating and Adding

Creating Dictionaries

- You'll need the using System.Collections.Generic; in the usings

- When creating a Dictionary, you must specify the type for the keys and the type for the value. NOTE: the types can be different.

```
Dictionary<WeaponType, int> inventory = new Dictionary<WeaponType, int>();
Dictionary<string, List<string>> wordDictionary = new Dictionary<string, List<string>>();
```

Adding to Dictionaries

- There are 3 ways to add items to a Dictionary

1. Add items in the initializer {key, value}

```
Dictionary<string, float> menu = new Dictionary<string, float>() {
    {"Hamburger", 0.99F }, //key,value Pair
    {"Cheeseburger", 1.50F }
};
```

2. Use the Add method. Add(key, value)
   NOTE: this can throw an exception. Before using, call ContainsKey first.

```
//Add method
menu.Add("Fries", 0.99F);
menu.Add("Soda", 0.99F);
```

3. Use [ ]. [key] = value

```
//[ ] -- add or update
//if nuggets is NOT in the dictionary, it will add it ELSE it will update it
menu["Chicken Nuggets"] = 2F;
```

Challenge #1: Creating and Adding

1. Create a Dictionary where the keys are string and the values are doubles and call the variable pg2.
2. Using the Random class, add 10 students with random grades (0-100) to the pg2 dictionary.

Examples:

```
Dictionary<string, float> menu = new Dictionary<string, float>() {
    {"Hamburger", 0.99F }, //key,value Pair
    {"Cheeseburger", 1.50F }
};

//Add method
menu.Add("Fries", 0.99F);
menu.Add("Soda", 0.99F);

//[ ] -- add or update
//if nuggets is NOT in the dictionary, it will add it ELSE it will update it
menu["Chicken Nuggets"] = 2F;
```

LINKS

Dictionary

Add

SLIDES

Creating How-To

Add How-To

Looping

Looping over Dictionaries

- foreach loops

```
foreach (KeyValuePair<string, float> menuItem in menu)
{
    Console.WriteLine($"{menuItem.Key}: {menuItem.Value:C2}");
}
```

Challenge #2: Printing

1. Create a method called PrintGrades and pass the dictionary as a parameter.
2. Loop over the dictionary and print the student names and grades.
3. Call PrintGrades from Main.

Example:

```
foreach (KeyValuePair<string, float> menuItem in menu)
{
    Console.WriteLine($"{menuItem.Key}: {menuItem.Value:C2}");
}
```

LINKS

foreach

Interpolated strings

Formatting

SLIDES

Looping How-To

Challenge #2: Printing

INTERMEDIATE LEVEL:

- Left-align the names and right-align the # (print only 2 decimal places)
- Color code the grades.
  - A: Green
    B: Dark Green
    C: Yellow
    D: Dark Yellow
    F: Red

LINKS

foreach

Interpolated strings

Formatting

SLIDES

Looping How-To

Removing

Removing from Dictionaries

- Use the Remove(key) method
  NOTE: it will return true/false if the key and value were removed.

```
bool isRemoved = menu.Remove("Chicken Nuggets");
```

Challenge #3: Removing

1. Create a method called DropStudent and pass the dictionary as a parameter.
2. Ask the user to enter a name to remove.
3. Remove the item from the pg2 dictionary.
4. Check the result and print a message.
   - If the key was found, print that the name was remove.
   - If the key was not found, print that the student was not in pg2.

Example:

```
bool isRemoved = menu.Remove("Chicken Nuggets");
```

LINKS

Remove

SLIDES

Removing How-To

Checking for Keys

Checking for Keys

- There are 2 ways to check for keys in a Dictionary. Both will return true/false if the key was found.

1. ContainsKey(key)

```
bool found = menu.ContainsKey("Chocolate Shake");
if (found) Console.WriteLine("Chocolate shake is on the menu.");
```

2. TryGetValue(key,out value)

```
string offMenu = "hash browns";
bool foundKey = menu.TryGetValue(offMenu, out float cost);
if(foundKey) Console.WriteLine($"{offMenu} Price: {cost}");
```

Getting Values

- There are 2 ways to get the value for a key.

1. Use TryGetValue(key,out value)

```
string offMenu = "hash browns";
bool foundKey = menu.TryGetValue(offMenu, out float cost);
if(foundKey) Console.WriteLine($"{offMenu} Price: {cost}");
```

2. Use [ ]

```
float menuPrice = menu["Hamburger"];
```

Updating Values

- To update a value for a key that is already in the dictionary:
  - Use [key]. Put the key inside the [ ]

```
menu["Hamburger"] = 4.99F;
```

Challenge #4: Updating a value

1. Create a method called CurveStudent and pass the dictionary as a parameter.
2. Ask the user to enter a name to curve.
3. Use TryGetValue to get the value.
4. If TryGetValue returns true, use the value returned to add +5 to the grade. Put the curved value back into dictionary.
5. If TryGetValue returns false, print an error message.
6. Call CurveStudent from Main.

Example:

```
string offMenu = "hash browns";
bool foundKey = menu.TryGetValue(offMenu, out float cost);
if(foundKey) Console.WriteLine($"{offMenu} Price: {cost}");
```

LINKS

TryGetValue

SLIDES

Checking Keys How-To