

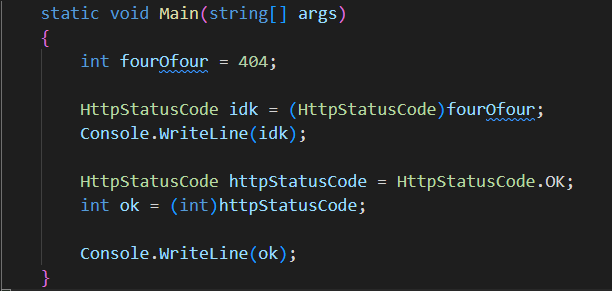
Stack.Push() Queue.Enqueue() dict.Add(k, v) jaggedArray (array.Length & array[i].Length)

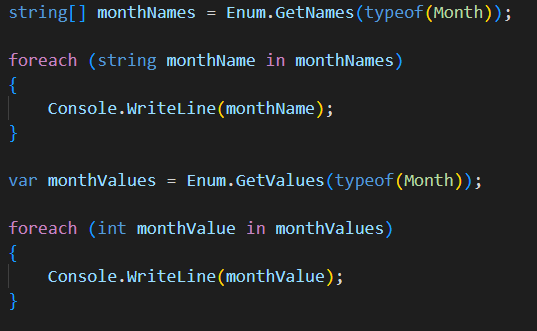
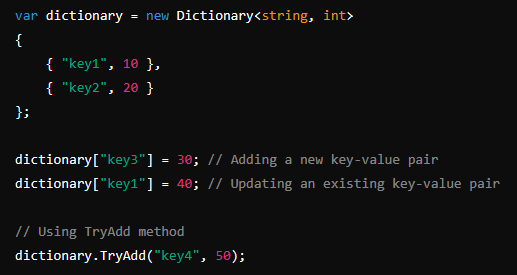
Stack.Peek() Queue.Peek() dict.ContainsKey() mdimensionalArray (array.GetLength(0/1))

Stack.Pop() Queue.Dequeue() dict.ContainsValue() dict/stack/queue.Count

if (other == null) return 1;!!! dict.Remove(key) : IComparable<Person>!!!!!





var tuple = Tuple.Create(1, "hello", 3.5); var valueTuple = (1, "hello", 3.5); (Mutable)

int firstItem = tuple.Item1; // 1 int firstItem = valueTuple.Item1; // 1

string secondItem = tuple.Item2; // "hello" string secondItem = valueTuple.Item2; // "hello"

double thirdItem = tuple.Item3; // 3.5 double thirdItem = valueTuple.Item3; // 3.



// Implementing IComparable interface

public int CompareTo(Person other)

{

this.(property).CompareTo(other.Age);

}

static EnrollmentSystem() // static constructor

{

foreach (Course course in Enum.GetValues(typeof(Course)))

{

WaitList.Add(course, new Queue<Student>());

EnrolledStudents.Add(course, new List<Student>());

}

}

var groupedRestaurants = from r in restaurants group r by r.OpeningYear into g orderby g.Count() descending select new { Year = g.Key, Count = g.Count() };

var combinedArray = persons.Union(students).ToArray();