List :

using System;

using System.Collections.Generic;

namespace L\_List

{

class Program

{

static void Main(string[] args)

{

List<int> a = new List<int>();

Random r = new Random();

PrintValues(a);

for (int i = 0; i < 10; i++)

a.Add(r.Next(100));

PrintValues(a);

a.Sort();

PrintValues(a);

a.RemoveAt(3);

PrintValues(a);

}

private static void PrintValues(List<int> a)

{

Console.WriteLine("Print Values in List<int>");

Console.WriteLine(" Count = {0}", a.Count);

Console.WriteLine(" Capacity = {0}", a.Capacity);

foreach (var i in a)

Console.Write(" {0}", i);

Console.WriteLine();

}

}

}

ArrayList :

using System;

using System.Collections;

namespace A\_ArrayList

{

class Program

{

static void Main(string[] args)

{

ArrayList a = new ArrayList();

Random r = new Random();

PrintValues(a);

for (int i = 0; i < 10; i++)

a.Add(r.Next(100));

PrintValues(a);

a.Sort();

PrintValues(a);

a.RemoveAt(3);

PrintValues(a);

}

private static void PrintValues(ArrayList a)

{

Console.WriteLine("Print Values in ArrayList");

Console.WriteLine(" Count = {0}", a.Count);

Console.WriteLine(" Capacity = {0}", a.Capacity);

foreach (var i in a)

Console.Write(" {0}", i);

Console.WriteLine();

}

}

}

Dictionary :

using System;

using System.Collections.Generic;

namespace D\_Dictionary

{

class Program

{

static void Main(string[] args)

{

Dictionary<string, string> colorTable = new Dictionary<string, string>();

colorTable.Add("Red", "빨간색");

colorTable.Add("Green", "초록색");

colorTable.Add("Blue", "파란색");

foreach(var v in colorTable)

Console.WriteLine("colorTable[{0}] = {1}", v.Key, v.Value);

Console.WriteLine();

Console.WriteLine("\n"+colorTable["Red"]);

Console.WriteLine(colorTable["Green"]);

Console.WriteLine(colorTable["Blue"]);

}

}

}

큐 :

using System;

using System.Collections.Generic;

using System.Collections;

namespace Q\_Queue

{

class Program

{

static void Main(string[] args)

{

Queue<string> que = new Queue<string>();

que.Enqueue("Tiger");

que.Enqueue("Lion");

que.Enqueue("Zebra");

que.Enqueue("Cow");

que.Enqueue("Rabbit");

PrintQueue("que: ", que);

Console.WriteLine(" Dequeuing '{0}'", que.Dequeue());

Console.WriteLine(" Peek: '{0}'", que.Peek());

Queue<string> que2 = new Queue<string>(que.ToArray());

PrintQueue("que2:", que2);

string[] array = new string[que.Count];

que.CopyTo(array, 0);

Queue<string> que3 = new Queue<string>(array);

PrintQueue("que3:", que3);

Console.WriteLine("que.Contains(Lion) = {0}", que.Contains("Lion"));

que3.Clear();

Console.WriteLine("Count = {0}, {1}, {2}", que.Count, que2.Count, que3.Count);

// 제너릭이 아닌 Queue

Queue myQ = new Queue();

myQ.Enqueue("one");

myQ.Enqueue("two");

myQ.Enqueue("three");

// Displays the properties and values of the Queue.

Console.WriteLine("myQ");

Console.WriteLine("\tCount: {0}", myQ.Count);

Console.Write("\tValues:");

PrintValues(myQ);

}

private static void PrintQueue(string s, Queue<string> que)

{

Console.Write("{0,-8}", s);

foreach(var item in que)

Console.Write("{0,-8}", item);

Console.WriteLine();

}

private static void PrintValues(Queue myQ)

{

foreach (string v in myQ)

Console.Write(" {0}", v);

Console.WriteLine();

}

public static void PrintValues(IEnumerable myCollection)

{

foreach (Object obj in myCollection)

Console.Write(" {0}", obj);

Console.WriteLine();

}

}

}

스택 :

using System;

using System.Collections.Generic;

namespace S\_Stack

{

class Program

{

static void Main(string[] args)

{

Console.Write("계산할 수식을 Polish 표기법으로 입력하세요: ");

string[] token = Console.ReadLine().Split();

foreach (var i in token)

Console.Write(" {0}", i);

Console.Write(" = ");

Stack<double> nStack = new Stack<double>();

foreach(var s in token)

{

if(isOperator(s))

{

switch (s)

{

case "+":

nStack.Push(nStack.Pop() + nStack.Pop());

break;

case "-":

nStack.Push(-(nStack.Pop() - nStack.Pop()));

break;

case "\*":

nStack.Push(nStack.Pop() \* nStack.Pop());

break;

case "/":

nStack.Push(1.0/(nStack.Pop() / nStack.Pop()));

break;

}

}

else

{

nStack.Push(double.Parse(s));

}

}

Console.WriteLine("결과는 {0}", nStack.Pop());

}

private static bool isOperator(string s)

{

if (s == "+" || s == "-" || s == "\*" || s == "/")

return true;

else

return false;

}

private static bool isNumber(string s)

{

if (Double.TryParse(s, out double number))

return true;

else

return false;

}

}

}