Array

: It’s a data structure that has elements of same type

Advantage : All the elements share common name

We can access elements thru their position(index)

The elements are stored in a contiguous location

No need to declare different elements

Syntax :

Data-type [] arrayname = new data-type[size];

Int[] num = new int[10];

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

Num[0] num[1] num[2] ………………………………………………………………………………………………….num[8]

Arrays could be

One Dimensional

2-D Array

N-D Array

Jagged Array

using System;

namespace FirstProject

{

class SingleDArrayDemo

{

static void Main()

{

int[] num = new int[10];

Console.WriteLine("Enter Elements");

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

{

num[i] = Convert.ToInt16(Console.ReadLine());

}

Console.WriteLine("Elements are");

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

{

Console.WriteLine(num[i]);

}

Console.WriteLine("Elements in reverse order are");

for (int i = 9; i >=0; i--)

{

Console.WriteLine(num[i]);

}

Console.Read();

}

}

}

Sum & Average of elements

using System;

namespace FirstProject

{

class SingleDArrayDemo

{

static void Main()

{

int[] num = new int[10];

Console.WriteLine("Enter Elements");

int sum = 0;

float avg;

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

{

num[i] = Convert.ToInt16(Console.ReadLine());

sum += num[i];

}

avg = (float)sum / 10;

Console.WriteLine("Sum is " + sum);

Console.WriteLine("Average is " + avg);

Console.Read();

}

}

}

Initialize Array

using System;

namespace FirstProject

{

class SingleDArrayDemo

{

static void Main()

{

// Declare & Initialize Array

**int[] num = new int[]{ 1,2,3,4,5,6};**

// Console.WriteLine("Enter Elements");

int sum = 0;

float avg;

for(int i=0;i<num.Length;i++)

{

// num[i] = Convert.ToInt16(Console.ReadLine());

sum += num[i];

}

avg = (float)sum / 10;

Console.WriteLine("Sum is " + sum);

Console.WriteLine("Average is " + avg);

Console.Read();

}

}

}

2-D Array (3 by 3 Array)

|  |  |  |
| --- | --- | --- |
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class \_2DArrayDemo

{

static void Main()

{

int[,] num = new int[3, 3];

Console.WriteLine("Enter Elements");

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

num[i, j] = Convert.ToInt16(Console.ReadLine())

; }

}

Console.WriteLine("Elements are");

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

Console.Write(num[i, j] + "\t");

}

Console.WriteLine();

}

}

}

}

Initialize 2-D Array

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class \_2DArrayDemo

{

static void Main()

{

int[,] num = new int[,]

{

{1,2,3 },

{1,2,3 },

{1,2,3 },

{1,2,3 }

};

Console.WriteLine("Enter Elements");

for(int i=0;i<4;i++)

{

for(int j=0;j<3;j++)

{

num[i, j] = Convert.ToInt16(Console.ReadLine())

; }

}

Console.WriteLine("Elements are");

for (int i = 0; i <4; i++)

{

for (int j = 0; j < 3; j++)

{

Console.Write(num[i, j] + "\t");

}

Console.WriteLine();

}

}

}

}

-----------------------------------

Jagged Array : An Array of Array

We want to store marks of 10 students in some subjects, but the no. of subjects are not same for every student

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class JaggedArrayDemo

{

static void Main()

{

int[][] jagged\_array = new int[10][];

jagged\_array[0] = new int[4];

Console.WriteLine("Enter Marks for Student 1");

for(int i=0;i<4;i++)

{

jagged\_array[0][i] = Convert.ToInt16(Console.ReadLine());

}

Console.WriteLine("Enter Marks for Student 2");

jagged\_array[1] = new int[2];

for (int i = 0; i < 2; i++)

{

jagged\_array[1][i] = Convert.ToInt16(Console.ReadLine());

}

}

}

}

// C# program to illustrate the declaration

// and Initialization of Jagged Arrays

using System;

class GFG {

// Main Method

public static void Main()

{

// Declare the Jagged Array of four elements:

int[][] jagged\_arr = new int[4][];

// Initialize the elements

jagged\_arr[0] = new int[] {1, 2, 3, 4};

jagged\_arr[1] = new int[] {11, 34, 67};

jagged\_arr[2] = new int[] {89, 23};

jagged\_arr[3] = new int[] {0, 45, 78, 53, 99};

// Display the array elements:

for (int n = 0; n < jagged\_arr.Length; n++) {

// Print the row number

System.Console.Write("Row({0}): ", n);

for (int k = 0; k < jagged\_arr[n].Length; k++) {

// Print the elements in the row

System.Console.Write("{0} ", jagged\_arr[n][k]);

}

System.Console.WriteLine();

}

}

}

**Advantage** :

Int[] num = new int[20];

15 bytes are wasted

Memory requirement shud be known at compile time

* All the elements share common name
* We can access elements thru their position(index) (1)
* The elements are stored in a contiguous location
* No need to declare different elements
* Similar data type

**Disadvantages of Arrays**

* Size is fixed , memory is static

Int[] num = new int[20];

* Insertion and deletion of elements is time consuming because it requires lots of reshuffling of elements

To solve these issues , we use Collections

Collection means that it’s a data structure wherein we can store hetrogeneous elements(of different types)

We have different classes for collection

These classes are present in which namespace (System.Collections)

* ArrayList
* Stack
* Queue
* Hashtable

Collection

Advantages

* Size is dynamic , memory is dynamic which means memory is not wasted
* Insertion and deletion of elements is easy. We can call inbuilt methods to do these operations

ArrayList

using System;

using System.Collections;

namespace FirstProject

{

class CollectionsDemo

{

static void Main()

{

// int[] num = new int[10];

ArrayList list = new ArrayList();

list.Add(1);

list.Add(2);

list.Add(100);

for(int i=0;i<list.Count;i++)

Console.WriteLine(list[i]);

list.Insert(0, 200);

list.Insert(2, 400);

Console.WriteLine("Elements after insertion");

for (int i = 0; i < list.Count; i++)

Console.WriteLine(list[i]);

list.RemoveAt(0);

list.Remove(1);

Console.WriteLine("Elements after deletion");

for (int i = 0; i < list.Count; i++)

Console.WriteLine(list[i]);

}

}

}

using System;

using System.Collections;

namespace FirstProject

{

class CollectionsDemo

{

static void Main()

{

// int[] num = new int[10];

ArrayList list = new ArrayList();

list.Add(1);

list.Add(2);

list.Add(100);

for(int i=0;i<list.Count;i++)

Console.WriteLine(list[i]);

list.Insert(0, 200);

list.Insert(2, 400);

Console.WriteLine("Elements after insertion");

for (int i = 0; i < list.Count; i++)

Console.WriteLine(list[i]);

list.RemoveAt(0);

list.Remove(1);

Console.WriteLine("Elements after deletion");

for (int i = 0; i < list.Count; i++)

Console.WriteLine(list[i]);

Console.WriteLine("Enter Element to search");

int x = Convert.ToInt16(Console.ReadLine());

for (int i = 0; i < list.Count; i++)

{

if ((int)list[i] == x)

{

Console.WriteLine("Element Found" + list[i]);

}

}

}

}

}

-------------------------------------------

In Arraylist > Insertion / deletion can be done from either side

Queue : Follows FIFO > First In , First Out > Insertion will be done at end , deletion will be done from beginning

using System;

using System.Collections;

namespace FirstProject

{

class QueueDemo

{

static void Main()

{

Queue queue = new Queue();

queue.Enqueue(1);

queue.Enqueue(2);

queue.Enqueue(3);

Console.WriteLine("ELements are ");

foreach(int x in queue)

Console.WriteLine(x);

queue.Dequeue();

Console.WriteLine("ELements after deletion are ");

foreach (int x in queue)

Console.WriteLine(x);

}

}

}

In Arraylist > Insertion / deletion can be done from either side

Queue : Follows FIFO > First In , First Out > Insertion will be done at end , deletion will be done from beginning

Stack : Follows LIFO > Last In , First Out > Insertion and deletion will be done at end only

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class StackDemo

{

static void Main()

{

Stack stack = new Stack();

stack.Push(10);

stack.Push(20);

stack.Push(30);

stack.Push(40);

Console.WriteLine("Elements are ");

foreach(int x in stack)

{

Console.WriteLine(x);

}

//for(int i=0;i<stack.Count;i++)

//{

// Console.WriteLine(stack.Peek());

//}

stack.Pop();

Console.WriteLine("Elements after Deletion are ");

foreach (int x in stack)

{

Console.WriteLine(x);

}

}

}

}

In Arraylist > Insertion / deletion can be done from either side

Queue : Follows FIFO > First In , First Out > Insertion will be done at end , deletion will be done from beginning

Stack : Follows LIFO > Last In , First Out > Insertion and deletion will be done at end only

In all these collection classes, we are accessing elements thru their position, in case we don’t want to access elements through their position, we use **Hashtable**

**Here entries are done**

**[index] = value , for arrays/ ArrayList, Stack, Queue, its like this**

**<key , value> , we use in HashTable**

[key] = value

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstProject

{

class HasSetDemo

{

static void Main()

{

Hashtable hashtable = new Hashtable();

hashtable[1] = 100;

hashtable[2] = 200;

hashtable[100] = 456;

hashtable[290] = 90;

Console.WriteLine("Marks are ");

for (int i=0;i<hashtable.Keys.Count;i++)

{

Console.WriteLine(hashtable[i]);

}

foreach (int x in hashtable.Keys)

{

Console.WriteLine(hashtable[x]);

}

Console.WriteLine(hashtable[100]);

Console.WriteLine(hashtable[290]);

}

}

}