using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

//string name;

//int age;

//string address;

//char gender;

////Console.WriteLine("Hello");

////Console.WriteLine("Bye");

//// Ctrl + KC > Comment Lines

//// Ctrl + KU > UnComment Lines

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

//name = Console.ReadLine();

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

//age = Convert.ToInt16(Console.ReadLine());

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

//address = Console.ReadLine();

Console.WriteLine("Enter Gender");

int gender1 = Console.Read();

Console.WriteLine(gender1);

// Console.WriteLine("Name is {0}\nAddress is{1}\nAge is{2}\nGernder is{3}",name,address,age,gender );

}

}

}

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

string name;

int age;

string address;

char gender;

//Console.WriteLine("Hello");

//Console.WriteLine("Bye");

// Ctrl + KC > Comment Lines

// Ctrl + KU > UnComment Lines

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Age");

age = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Enter Address");

address = Console.ReadLine();

Console.WriteLine("Enter Gender");

gender = Convert.ToChar(Console.Read());

Console.WriteLine("Name is {0}\nAddress is {1}\nAge is {2}\nGender is {3}",name,address,age,gender );

}

}

}

To Take input from user

ReadLine() > returns input in string form

Read() > returns input in ASCII form, reads only one character

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

int num1, num2, res;

Console.WriteLine("Enter number1");

num1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter number2");

num2 = Convert.ToInt32(Console.ReadLine());

res = num1 + num2;

Console.WriteLine("Sum of {0} and {1} is {2}", num1,num2, res);

}

}

}

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

int num1, num2, res;

Console.WriteLine("Enter number1");

num1 = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter number2");

num2 = Convert.ToByte(Console.ReadLine());

res = num1 + num2;

Console.WriteLine("Sum of {0} and {1} is {2}", num1,num2, res);

}

}

}

Int x; > Allocates some bytes

Convert.ToInt16 > 16 bits 1 bit is reserved for sign 2^15 – 32768 to + 32767

Convert.ToInt32 > 32 bits 2^32 > Range is more

Convert.ToByte > 8 bits 2^8 > 256 0 -255

Convert.ToUInt16 > 16 bits No bit is reserved for sign , all bytes are reserved for numbers,

2^16 0 to

Int > 2 byes

Short int >

Console is a class

ReadLine() , Read() , Write () , WriteLine() are methods which belongs to Console Class

Console class belongs to System namespace

**Convert is a class**

**ToInt16, ToInt32, ToByte are methods**

static void Main(string[] args)

{

int num1, num2, res;

Console.WriteLine("Enter number1");

num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter number2");

num2 = Int16.Parse(Console.ReadLine());

res = num1 + num2;

Console.WriteLine("Sum of {0} and {1} is {2}", num1,num2, res);

}

static void Main(string[] args)

{

int num1, num2, res;

Console.WriteLine("Enter number1");

num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter number2");

num2 = byte.Parse(Console.ReadLine());

res = num1 + num2;

Console.WriteLine("Sum of {0} and {1} is {2}", num1,num2, res);

}

Function : Set of statements which perform a specific task

Advantage :

Resuablity

Modularity : Breaking a big program into smaller subprograms

Its easier to understand a small program

Its easier to debug a small program

Its easier to mantain a small program

How do we make a function

Return\_type function\_name(parameters)

{

Statements;

}

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void add(int x, int y)

{

Console.WriteLine("Sum of {0} and {1} is {2} " , x, y, (x+y) );

}

static void subtract(int x, int y)

{

Console.WriteLine("Difference of {0} and {1} is {2} ", x, y, (x - y));

}

static void product(int x, int y)

{

Console.WriteLine("Product of {0} and {1} is {2} ", x, y, (x \* y));

}

static void quotient(int x, int y)

{

Console.WriteLine("Quotient of {0} and {1} is {2} ", x, y, (x / y));

}

static void Main(string[] args)

{

int x, y;

Console.WriteLine("ENter Value of x");

x = Convert.ToByte(Console.ReadLine());

Console.WriteLine("ENter Value of y");

y = Convert.ToByte(Console.ReadLine());

add(x, y);

subtract(x, y);

product(x, y);

quotient(x, y);

}

}

}

-------

using System; // namespace

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static int add(int x, int y)

{

return x + y;

}

static int subtract(int x, int y)

{

return x-y;

}

static int product(int x, int y)

{

return x \* y;

}

static float quotient(int x, int y)

{

return x / y;

}

static void Main(string[] args)

{

int x, y;

int res;

Console.Clear(); // To clear Screen

Console.WriteLine("ENter Value of x");

x = Convert.ToByte(Console.ReadLine());

Console.WriteLine("ENter Value of y");

y = Convert.ToByte(Console.ReadLine());

res = add(x, y);

Console.WriteLine("Sum of {0} and {1} is {2} ", x, y, res);

res = subtract(x, y);

Console.WriteLine("Difference of {0} and {1} is {2} ", x, y, res);

res=product(x, y);

Console.WriteLine("Product of {0} and {1} is {2} ", x, y, res);

float res1 = quotient(x, y);

Console.WriteLine("Quotient of {0} and {1} is {2} ", x, y, res1);

Console.Read();

}

}

}

Method Overloading : Functions having same name, but different no. or type of parameters

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Method\_Overloading

{

public static int add(int x, int y)

{

return x + y;

}

public static int add2(int x, int y, int z)

{

return x + y +z;

}

public static int add3(int x, int y, int z,int a)

{

return x + y + z + a;

}

public static float add4(float x, float y)

{

return x + y;

}

public static void Main()

{

int x = 10;

int y = 2;

Console.WriteLine("Sum is " + add(1, 2));

Console.WriteLine("Sum is " + add3(2,3,4,4));

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Method\_Overloading

{

public static int add(int x, int y)

{

return x + y;

}

public static int add(int x, int y, int z)

{

return x + y +z;

}

public static int add(int x, int y, int z,int a)

{

return x + y + z + a;

}

public static float add(float x, float y)

{

return x + y;

}

public static void Main()

{

int x = 10;

int y = 2;

Console.WriteLine("Sum is " + add(1, 2));

Console.WriteLine("Sum is " + add(2,3,4,4));

// ConsoleApp1.Program.Main();

}

}

}

// params array

Are used when you want to pass any no. of values of one type to a function

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Method\_Overloading

{

public static int add(params int[] num)

{

int sum = 0;

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

sum += num[i];

return sum;

}

//public static int add(int x, int y)

//{

// return x + y;

//}

////public static void add(int x, int y)

////{

//// return x + y;

////}

public static float add(float x, float y)

{

return x + y;

}

//public static int add(int x, int y, int z)

//{

// return x + y +z;

//}

//public static int add(int x, int y, int z,int a)

//{

// return x + y + z + a;

//}

public static void Main()

{

int x = 10;

int y = 2;

Console.WriteLine("Sum is " + add(1, 2));

Console.WriteLine("Sum is " + add(2,3,4,4));

Console.WriteLine(add(1,1,1,1,1,1,1,1,1,2,3,4,4));

Console.WriteLine(add(1));

Console.WriteLine(add(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1));

// ConsoleApp1.Program.Main();

}

}

}

Output Parameters : are used to return more than one value from a function

public static void AllOperations(int x, int y, out int add , out int subt, out int product, out int div)

{

add = x + y;

subt = x - y; ;

div = x/y;

product = x\*y;

}

Call By Value : We pas the value of the variable to the method, which means it creates a local variable inside the called function. Inside that function, whatever changes we do are left there only. Changes are not sent to the calling function

Call By Reference : We pas the address of the variable to the method, which means it does not creates a local variable inside the called function. Inside that function, whatever changes we do are sent to the calling function

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class CallByValue\_CallByReferenec

{

static void Change1(int x)

{

x = 200;

Console.WriteLine("VAlue of X in Change1 is " + x);

}

static void Change2(ref int x)

{

x = 300;

Console.WriteLine("VAlue of X in Change2 is " + x);

}

static void Main()

{

int X = 100;

Console.WriteLine("Value of X before calling Change1 is " + X);

Change1(X);

Console.WriteLine("Value of X after calling Change1 is " + X);

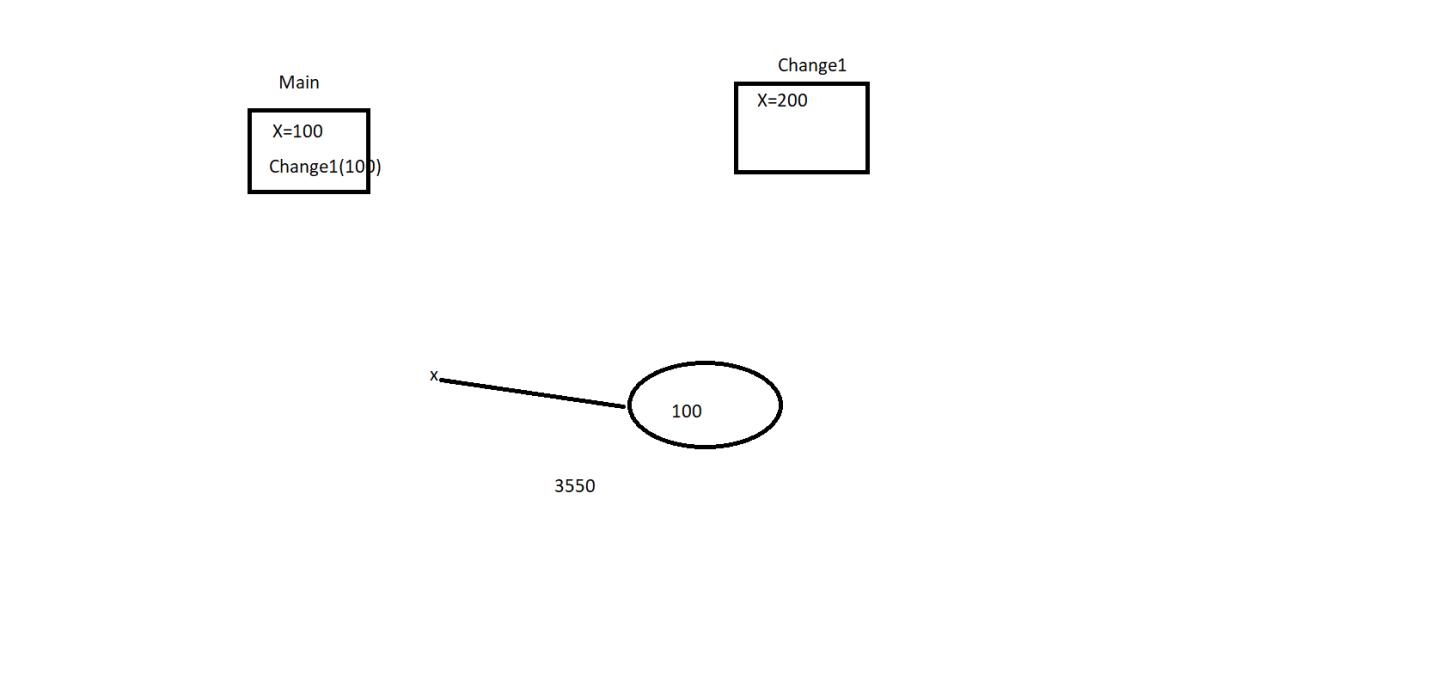
Change2(ref X);

Console.WriteLine("Value of X after calling Change2 is " + X);

}

}

}



Named Parameters

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class NamedPara\_DefaultPara

{

static void GetDetails( string name, string address, int id, string dept)

{

Console.WriteLine("Name is " + name);

Console.WriteLine("Address is " + address );

Console.WriteLine("Dept is "+ dept);

Console.WriteLine("ID is " + id);

}

static void Main()

{

GetDetails("Ajay", "Delhi", 12, "HR");

// Named Parameters while passing parameters to methids , we add their names

// We understand

GetDetails(name: "Ajay", id: 1, address: "Delhi", dept: "HR");

// Insert into emp (id, name , address) values (1,'Ajay','Delhi')

}

}

}

Default Parameters

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class NamedPara\_DefaultPara

{

static void GetDetails( string name, int id, string address ="Delhi" ,string dept="HR")

{

Console.WriteLine("Name is " + name);

Console.WriteLine("Address is " + address );

Console.WriteLine("Dept is "+ dept);

Console.WriteLine("ID is " + id);

}

static void Main()

{

GetDetails("Jatin", 2);

GetDetails("Ajay", 12, "Delhi", "HR");

// Named Parameters while passing parameters to methids , we add their names

// We understand

GetDetails(name: "Ajay", id: 1, address: "Delhi");

// Insert into emp (id, name , address) values (1,'Ajay','Delhi')

// GetDetails()

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class CallingValues

{

static int add(int x, int y, out int add, out int subt)

{

add = x + y;

subt = x - y;

return x + y;

}

static void Main()

{

int add1, subt2;

int res= add(10,20, out add1, out subt2);

Console.WriteLine(res + 10 + add1);

}

}

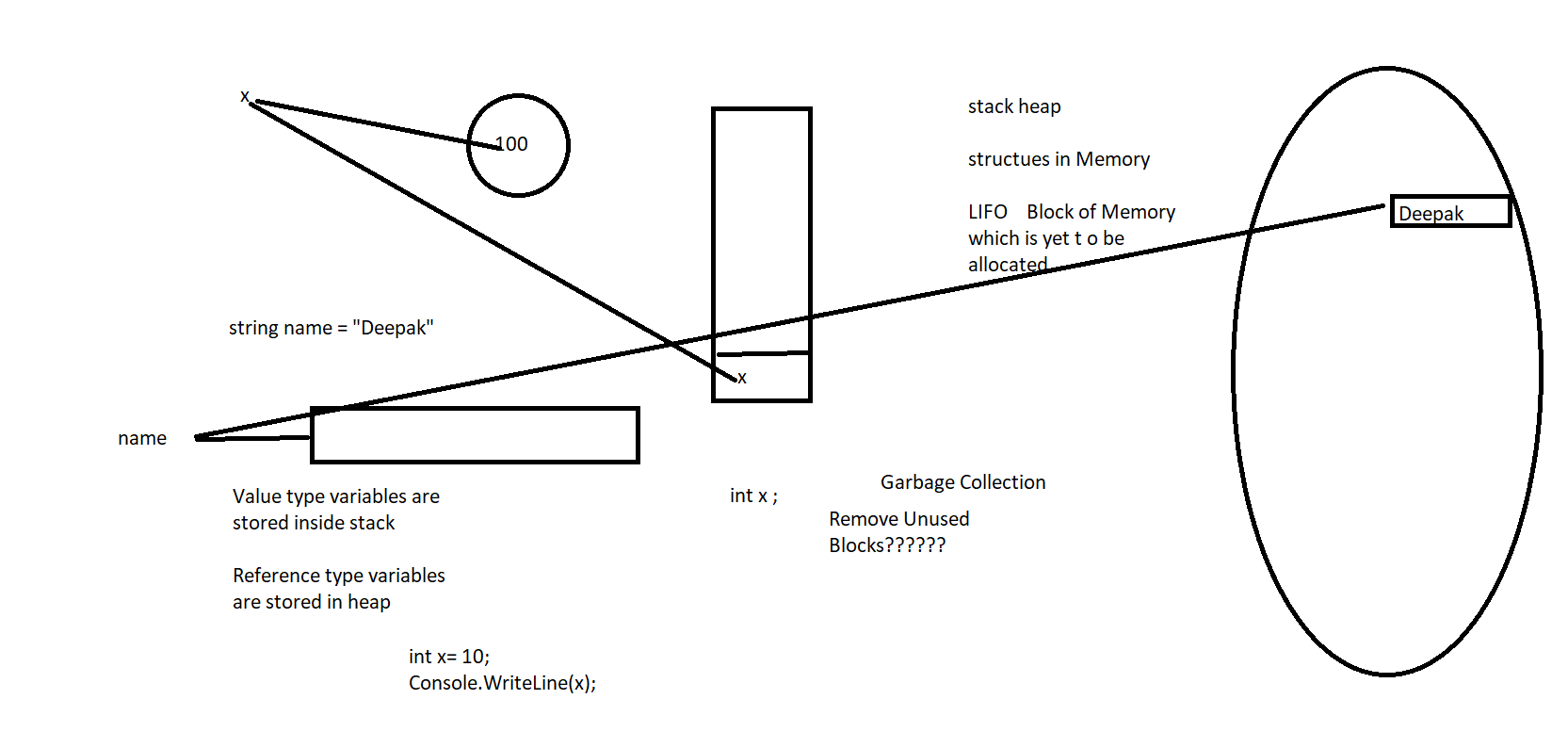
}

Variables

Value type & Reference Types

Variables which directly stores values , int , char , float Boolean struct enum

Int x = 100;



Reference type variables : which points to address where variable are stored

Ex : string class array

Refer type variables are stored on heap : Garbage Collection is done for variables which are stored on heap

// String and StringBuilder

String is immutable : It does not allows any changed or modifications

String name = “Ajay”;

Name + = “Kumar”;

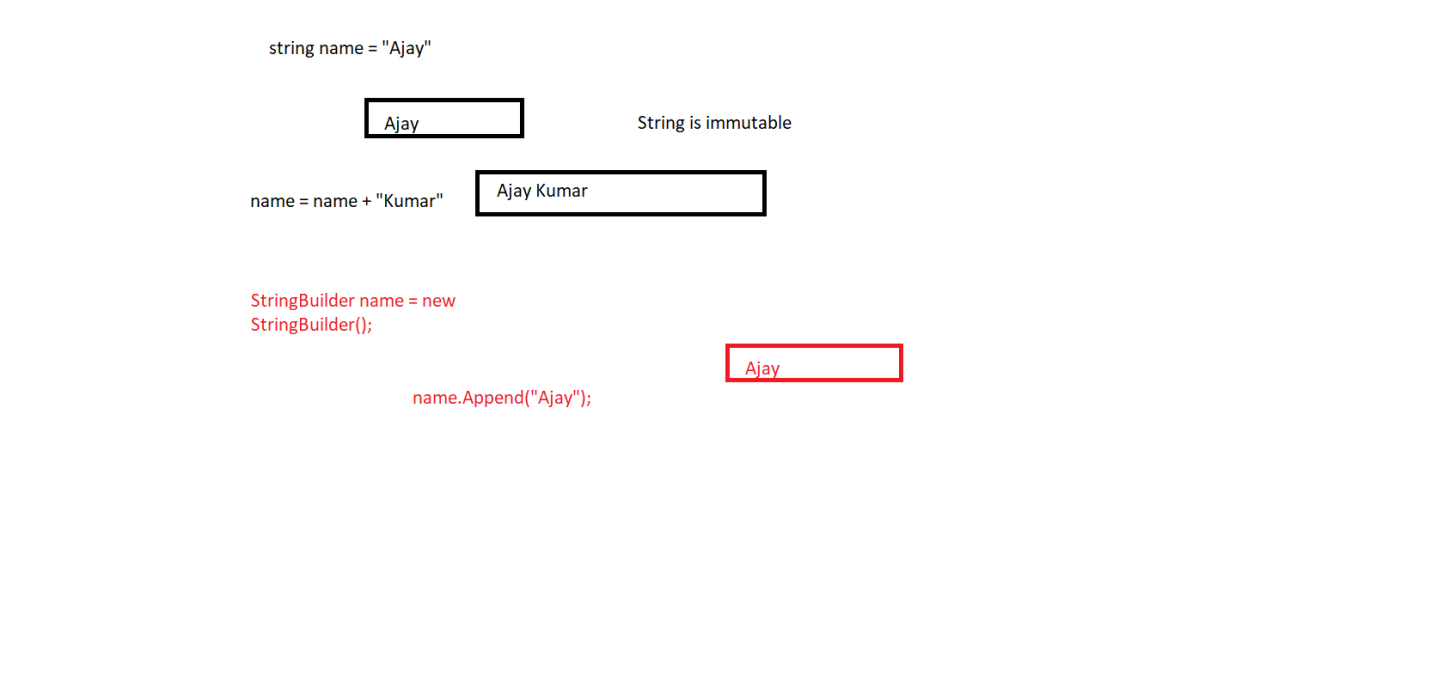
// It means that it will take more time

StringBuilder Class is Mutable : It allows changes

StringBulider name = new StringBuilder();

Name.Append(“Ajay”);

Name.Append(“Kuamar”);



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SecondDemo

{

class Program

{

static void Main(string[] args)

{

string name = "Ajay";

name += "Kumar";

StringBuilder str = new StringBuilder();

str.Append("Ajay");

str.Append("Kumar");

}

}

}

// Array is a collection of elements of same type

Declare 3 nos

Declare 100 elements

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SecondDemo

{

class Program

{

static void Main(string[] args)

{

// Array

int[] numArray = new int[10];

Console.WriteLine("ENter Elemenmts");

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

numArray[i] = int.Parse(Console.ReadLine());

Console.WriteLine("Elements are");

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

Console.WriteLine(numArray[i]);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SecondDemo

{

class Program

{

static void Main(string[] args)

{

// Array

int[] numArray = new[] { 1, 2, 3, 4, };

//Console.WriteLine("ENter Elemenmts");

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

// numArray[i] = int.Parse(Console.ReadLine());

int sum = 0;

Console.WriteLine("Elements are");

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

{

sum += numArray[i];

}

float avg = sum / numArray.Length;

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

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

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

for (int i = numArray.Length-1; i >=0; i--)

{

Console.WriteLine(numArray[i]);

}

}

}

}

2 – D array

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SecondDemo

{

class Program

{

static void Main(string[] args)

{

// Array

int[,] numArray = new int[3, 3]

{

{ 1, 2, 3 },

{ 2,3,4},

{ 3,4,5}

};

Console.WriteLine("ENter Elemenmts");

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

{

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

{

numArray[i, j] = Convert.ToByte(Console.ReadLine());

}

}

Console.WriteLine("Elements are");

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

{

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

{

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

}

Console.WriteLine();

}

Console.Read();

}

}

}

* Jagged Array
* Array of Array
* When no. of columns are not fixed for every row

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SecondDemo

{

class Program

{

static void Main(string[] args)

{

// Jagged Array

int[][] numArray = new int[10][];

Console.WriteLine("ENter Elemenmts");

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

//{

numArray[0] = new int[3];

numArray[1] = new int[5];

// }

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

{

for (int j = 0; j < numArray[i].Length; i++)

{

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

}

}

Console.Read();

}

}

* }