**Method Overloading: Used to achieve polymorphism. Advantage : no need to remember so many method names.**

**Output parameters: A function returns more than one value using output parameters**

static void DoAllOperations(int x, int y ,out int add,

out int subtract, out int product, out float div)

{

add = x + y;

subtract = x - y;

product = x \* y;

div = (float)x / (float)y;

}

static void Main()

{

int add, subtract, product;

float div;

DoAllOperations(20, 10, out add, out subtract, out product,

out div);

Console.WriteLine("Addition is " + add);

Console.WriteLine("Subtraction is " + subtract);

Console.WriteLine("Product is " + product);

Console.WriteLine("Divsion is " + div);

**}**

**Named parameter :** While calling a method , when we pass parameter’ values , we pass it alongwith parameter name

Advantage : We can change sequence

It makes program easy to understand

**Optional parameter :** While defining method, we give parameters some default values , so If we don’t provide value to those parameters, they will take that default value

We have to provide values from RIGHT TO LEFT

**Call By Value : We pass value of the variable to the called method (default way)**

Int x = 10;

Int y =20;

Add(x,y);

**Cal By reference : We pass address of the variable to the called method**

using System;

class Prog

{

static void DisplayDetails(int rn , string name, int marks, string subject)

{

Console.WriteLine("Rn is " + rn);

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

Console.WriteLine("Subject is " + subject);

Console.WriteLine("Marks are " + marks);

}

static void SimpleInterest(int principal, int rate=5 , int time=10)

{

Console.WriteLine((principal\* rate\* time)/100);

}

//static void SimpleInterest(int principal=12000, int rate =9, int time = 10)

//{

// Console.WriteLine((principal \* rate \* time) / 100);

//}

static void DoAllOperations(int x, int y ,out int add,

out int subtract, out int product, out float div)

{

add = x + y;

subtract = x - y;

product = x \* y;

div = (float)x / (float)y;

}

static void Main()

{

SimpleInterest(rate: 9, principal: 9000, time: 2);

SimpleInterest(12000, 9);

SimpleInterest(12000);

SimpleInterest(12000, 2, 9);

DisplayDetails(19, "Ajay", 1, "Hindi");

//Named Parameter

DisplayDetails(marks: 12, name: "Ajay", subject: "Hindi", rn: 9);

int add, subtract, product;

float div;

DoAllOperations(20, 10, out add, out subtract, out product,

out div);

Console.WriteLine("Addition is " + add);

Console.WriteLine("Subtraction is " + subtract);

Console.WriteLine("Product is " + product);

Console.WriteLine("Divsion is " + div);

}

}

using System;

class Prog

{

static void change1(int x)

{

x = 100;

Console.WriteLine("Value of x in Change1 " + x);

}

static void change2(ref int x)

{

x = 200;

Console.WriteLine("Value of x in Change2 " + x);

}

static void Main()

{

int x = 10;

Console.WriteLine("Value of x in Main before calling change1" +

" is " + x);

change1(x);

Console.WriteLine("Value of x in Main after calling change1 " +

"is " + x);

change2(ref x);

Console.WriteLine("Value of x in Main after calling change2 " +

"is " + x);

}

}

Call by value > Value goes from one function to other , that value is stored in a local variable created in that called function, changes done in called function are nor reflected in the calling function

Call by reference > Reference or Address goes from one function to other , that variable is passed in that called function, changes done in called function are reflected in the calling function

int num1 , num2 , num3 ;

int num1 , num2 , num3 num4 ……………………………………………….. num50;

When we have to declare multiple variables to store same type of value , we can use Array

Array : It’s a structure which is used to store elements of same type/ It’s a contiguous allocated memory

int num1 , num2 , num3 ;

type[] name = new type[size];

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

Num[0] Num[1] Num[2] Num[9]

int[] num = new int[10];

Arrays

using System;

class Prog

{

static void Main()

{

int[] num = new int[10];

Console.WriteLine("Enter Numbers");

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

{

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

}

Console.WriteLine("Numbers are");

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

{

Console.WriteLine(num[i]);

}

}

}

using System;

class Prog

{

static void Main()

{

int[] num = new int[10];

Console.WriteLine("Enter Numbers");

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

{

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

}

Console.WriteLine("Numbers are");

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

{

Console.WriteLine(num[i]);

}

}

}

using System;

class Prog

{

// Dispaly Sum and Average of elements of an array

static void Main()

{

int[] num = new int[10];

int sum=0;

float avg;

Console.WriteLine("Enter Numbers");

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.WriteLine(Int64.MaxValue);

Console.WriteLine(Byte.MaxValue);

Console.WriteLine(long.MaxValue);

Declare & Initialize an Array

using System;

class Prog

{

// Dispaly Sum and Average of elements of an array

static void Main()

{

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

int sum = 0;

float avg;

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

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);

}

}

using System;

class Prog

{

// Dispaly Sum and Average of elements of an array

static void Main()

{

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

Console.WriteLine("Enter Numbers");

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

{

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

{

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

}

}

Console.WriteLine("Numbers are");

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

{

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

{

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

}

Console.WriteLine();

}

}

}

using System;

class Prog

{

// Dispaly Sum and Average of elements of an array

static void Main()

{

int[,] num = new int[,] {

{ 1,2,3},

{ 1,2,3},

{ 1,2,3}

};

Console.WriteLine("Enter Numbers");

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

{

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

{

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

}

}

Console.WriteLine("Numbers are");

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

{

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

{

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

}

Console.WriteLine();

}

}

}