Functions / Methods

Functions> are block of code which performs some specific task

Advantages

* Reusability
* Modularity
* Structured Way
* Easy to use
* Easy to debug
* Easy to maintain
* Easy to understand
* Easy to test

How to define function

Return\_type functionname(paramteres)

{

Statements;

Return value;

}

We will talk about

1. Method Overloading
2. Call By Value, Call By Reference
3. Output Parameters
4. Named Parameters
5. Default / Optional Parameters
6. Params Array

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MtFirstProject

{

class FunctionsDemo

{

static void AddNumbers1(int x, int y)

{

Console.WriteLine(x + y);

}

static void AddNumbers2(int x, int y, int z)

{

Console.WriteLine(x + y + z);

}

static void AddNumbers3(float x, float y)

{

Console.WriteLine(x + y);

}

static void Main()

{

int x = 10;

int y = 10;

AddNumbers(x, y);

}

}

}

Method Overloading > Overloading a method with different number and type of parameters

It actually is Polymorphism

Polymorphism > One name, different forms

Polymorphism is of 2 types

1. Compile Time > Achieved thru
   1. Method Overloading
   2. Operator Overloading
2. Run Time Polymorphism, also known as Dynamic Polymorphism > Achieved thru Method Overrdiding, Possible in Inheritance

**Named Parameters** : While calling method, when you pass parameters, we are passing them using names

It allows you to change sequence, also it makes program user friendly for developers;

**Output Parameters**

By default , a functions returns how many values = 1

But if you want that a function should return more than one values, Output Parameters are used for this

static void Calculation(int x, int y, out int add, out int subt,

out int product, out int quotient)

{

add = x + y;

subt = x - y;

product = x \* y;

quotient = x / y;

}

// Calling Part

int sum, subtract, product;

float remainder;

Calculation(20, 10, out sum,

out subtract,

out product,

out remainder);

Console.WriteLine($"Sum is {sum}");

Console.WriteLine($"Difference is {subtract}");

Console.WriteLine($"Product is {product}");

Console.WriteLine($"Remainder is {remainder}");

Optional / Default parameters

// Default / Optional Parameters

static void SI(int p=12000, int rate=9, int time=10)

{

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

}

// They can be called like this

SI(10000, 2, 7);

SI(100000, 2);

SI();

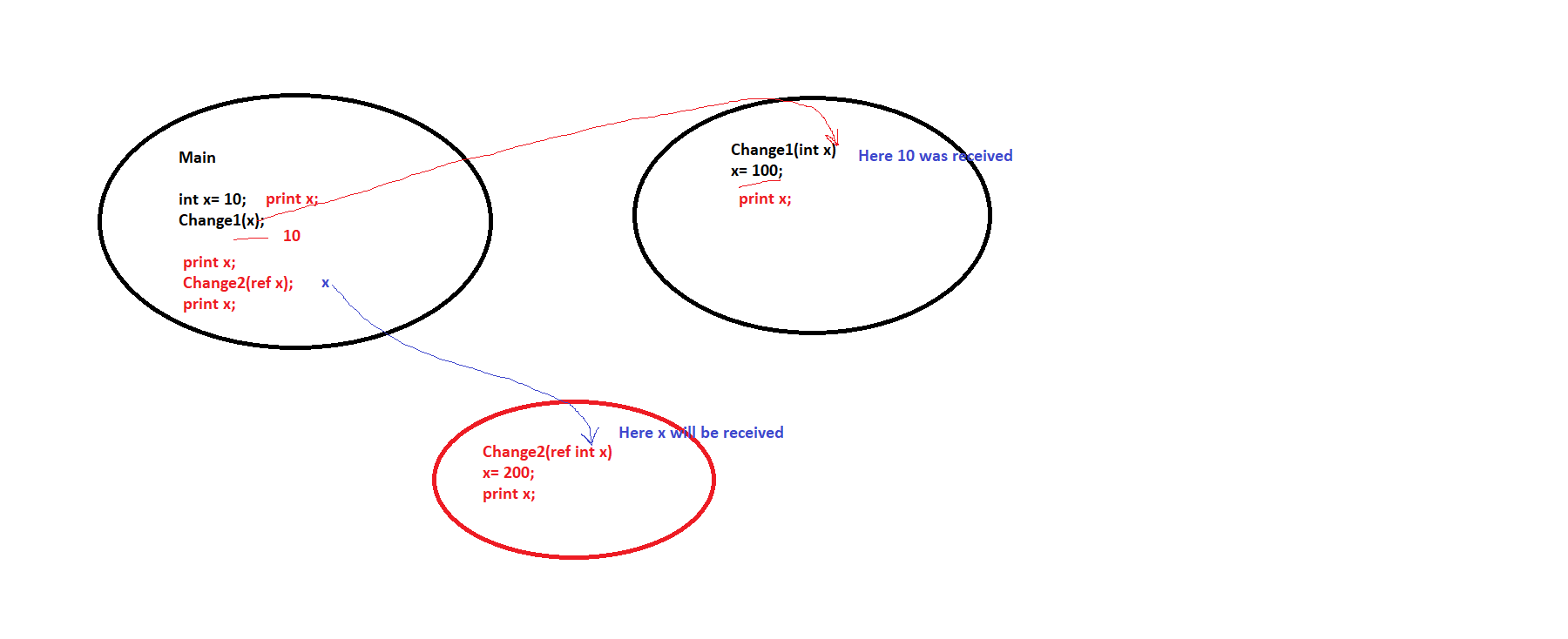
SI(13000);

SI(rate: 8);

Call By Value / Call By Reference

Call By Value > The value is passed

Call By Reference > The variable is passed



int X = 10;

Console.WriteLine("Value of X in Main Method is " + X); ;

Change1(X);

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

Change2(ref X);

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

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

static void Change1(int X)

{

Console.Write("Inside Change1 ");

X = 100;

Console.WriteLine(X);

}

static void Change2(ref int X)

{

Console.Write("Inside Change2");

X = 100;

Console.WriteLine(X);

}