1. **Anonymous Function:**

Anonymous function is a function inline statement used wherever the delegate type is expected.It is useful when the user wants to create an Inline function(A function that is expanded in line when it is called.) and anonymous function expects a parameter to be passed and can be converted to delegates.In which the delegates gets invoked using anonymous function.

Eg: for a class employee

delegate(Employee emp)

{

return emp.id == 3;

}

1. **Func< > and Action< >:**

Both Func and Action are the built-in generic types of delegate in which it is one of the Systems namespace collection

**Func :** It is a delegate that points to method which accepts one or more parameters and returns one out parameter

Syntax : Func< in parameter, out parameter> FuncName;

Eg:

class Program

{

static int Sum(int x, int y)

{

return x + y;

}

static void Main(string[] args)

{

Func<int,int, int> add = Sum;

int result = add(10, 10);

Console.WriteLine(result);

}

}

Output: 20

**Action:** It is also same as Func delegate the only difference between Func and Action is that the action will take parameter but

Won’t return a value in which the return type of action is void.

Syntax : Action<Data-type> ActionName;

Eg:

class Program

{

public delegate void Print(int val);

static void ConsolePrint(int i)

{

Console.WriteLine(i);

}

Static void Main(string[] args)

{

Action<int> printActionDel = ConsolePrint;

printActionDel(10);

}

}

Output : 10

1. **Predicate<> :**

It is also a built-in generic delegate function which returns either True or False in which It checks whether it matches the passed parameter or not. Predicate takes one input parameter and returns a value as bool(True or False).

Syntax: public delegate bool Predicate<in Parameter>(parameter object)

Eg:

class Program

{

static bool IsUpperCase(string str)

{

return str.Equals(str.ToUpper());

}

static void Main(string[] args)

{

Predicate<string> isUpper = IsUpperCase;

bool result = isUpper("hello world!!");

Console.WriteLine(result);

}

}

Output: False