Λ (λ) Expressions

# Short Introduction

Lambda expressions are **anonymous functions** that contain **expressions** or **sequence of operators**. All lambda expressions use the **lambda operator =>,** that can be read as “goes to” or “becomes”.

The left side of the lambda operator specifies the input parameters and the right side holds an expression or a code block that works with the entry parameters. Usually lambda expressions are used as predicates or instead of delegates.

**Format:** (input parameters) => Expression;

Parameter => expression  
Parameter-list => expression

**Expression examples**

Count => count + 2;Sum => sum + 2;n => n % 2 == 0

# Types of Lambda

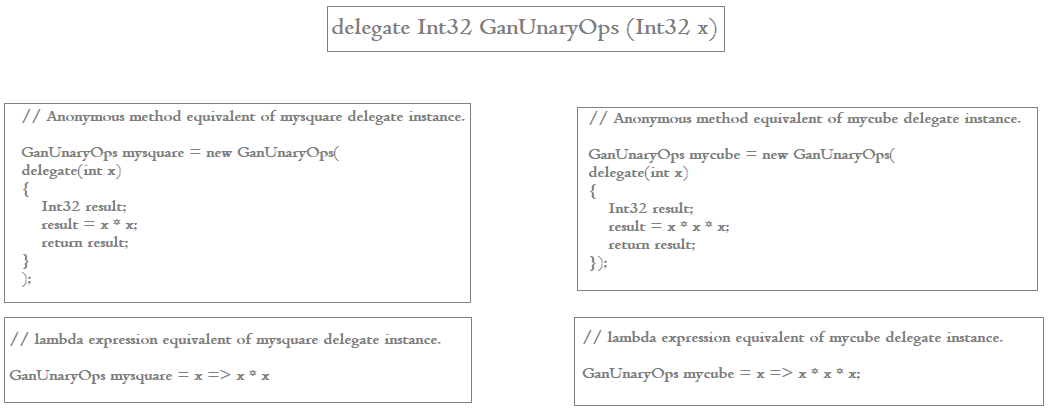
* **Expression Lambda** – just a single expression on the right hand side.

(input-parameters) => expression

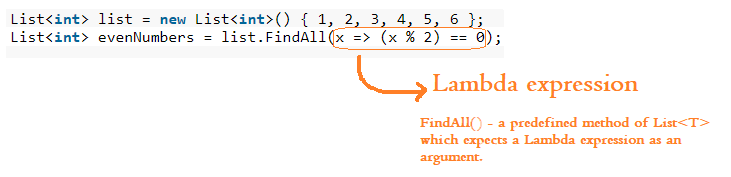
* **Statement Lambda** - a set of statements (more than one line enclosed by {}).

(input-parameters) => { <sequence-of-statements> }

# Lambda can be replacement for Anonymous methods



# Lambda can be passed as an expression to methods



# Lambda expression conversion to Delegate type

Any lambda expression can be converted to a [delegate](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/reference-types#the-delegate-type) type.  The delegate type to which a lambda expression can be converted is defined by the types of its parameters and return value. If a lambda expression doesn't return a value, it can be converted to one of the **Action delegate** types; otherwise, it can be converted to one of the**Func delegate** types.

For example, a lambda expression that has two parameters and returns no value can be converted to an [Action<T1,T2>](https://docs.microsoft.com/en-us/dotnet/api/system.action-2) delegate. A lambda expression that has one parameter and returns a value can be converted to a [Func<T,TResult>](https://docs.microsoft.com/en-us/dotnet/api/system.func-2) delegate.

In the following example, the lambda expression x => x \* x, which specifies a parameter that's named x and returns the value of x squared, is assigned to a variable of a delegate type:

Func<int, int> square = x => x \* x;

Console.WriteLine(square(5));

// Output:

// 25

# Automatic Inference of Types