```
namespace MMP.Tools
{
    public abstract class LinSpace<T>
    {
        protected T dt;
        T[] data;
        protected abstract T[] generate(T a, T b, int count, out T dt);

        protected LinSpace(T a, T b, int count)
        {
            data = generate(a, b, count, out dt);
        }

        public static implicit operator T[] (LinSpace<T> lsp)
        {
            return lsp.data;
        }

        public T this[int i] { get { return data[i]; } }
        public int Length { get { return data.Length; } }
        public T Distance { get { return dt; } }
        public T Left { get { return data[0]; } }
        public T Right { get { return data[data.Length - 1]; } }
    }
}


namespace MMP.Tools.Double
{
    using Real = System.Double;
    public class LinSpace : MMP.Tools.LinSpace<Real>
    {
        public LinSpace(Real a, Real b, int count = 100) : base(a, b, count)
        {
        }

        protected override Real[] generate(Real a, Real b, int count, out Real dt)
        {
            Real[] t = new Real[count];
            count--;
            dt = (b - a) / count;
            t[0] = a;
            for (int i = 1; i < count; i++) t[i] = t[i - 1] + dt;
            t[count] = b;
            return t;
        }
    }
}

namespace MMP.Tools.Single
{
    using Real = System.Single;
    public class LinSpace : MMP.Tools.LinSpace<Real>
    {
        public LinSpace(Real a, Real b, int count = 100) : base(a, b, count)
        {
        }

        protected override Real[] generate(Real a, Real b, int count, out Real dt)
        {
            Real[] t = new Real[count];
            count--;
            dt = (b - a) / count;
            t[0] = a;
            for (int i = 1; i < count; i++) t[i] = t[i - 1] + dt;
            t[count] = b;
            return t;
        }
    }
}


using System;

namespace MMP.Tools
{
    public static class Vectorize
    {
        public static T2[] Apply<T1, T2>(Func<T1, T2> f, T1[] x)
        {
            int n = x.Length;
            T2[] y = new T2[n];
            for (int i = 0; i < n; i++)
            {
                y[i] = f(x[i]);
            }
            return y;
        }

        public static T2[] Apply<T1, T2>(Func<T1, T2> f, LinSpace<T1> x)
        {
            int n = x.Length;
            T2[] y = new T2[n];
            for (int i = 0; i < n; i++)
            {
                y[i] = f(x[i]);
            }
            return y;
        }
    }
}


using System;
using MMP.Tools;
using static MMP.Tools.Vectorize;

namespace L5
{
    class Program
    {
        static void Main(string[] args)
        {
            LinSpace<double> x = new MMP.Tools.Double.LinSpace(0, 1);
            LinSpace<float> y = new MMP.Tools.Single.LinSpace(0, 1);
            float[] data = y;
            double[] sx = Apply(Math.Sin, x);
        }
    }
}
```