**16** Show that

$$a(v_h, w_h) \;=\; (K_h\, \underline{v}_h,\, \underline{w}_h)_{\ell^2}$$

where $K_h$ is the stiffness matrix (defined according to the lecture), $v_h,\, w_h \in V_{0h}$ (arbitrary), $\underline{v}_h$, $\underline{w}_h$ are the corresponding vectors according to the Ritz isomorphism, and $(\cdot,\cdot)_{\ell^2}$ is the Euclidean inner product. Show also that for the load vector $\underline{f}_h$,

$$\langle \widehat{F},\, v_h \rangle \;=\; (\underline{f}_h,\, \underline{v}_h)_{\ell^2}\,.$$

**17** Construct quadratic basis functions $\hat{\varphi}_0,\ \hat{\varphi}_1$ and $\hat{\varphi}_2 \in \mathcal{P}^2$ on the reference element $\hat{T} = [0,1]$ such that for $\xi_0 = 0,\ \xi_1 = 0.5,\ \xi_2 = 1$,

$$\hat{\varphi}_i(\xi_j) = \delta_{i,j}.$$

Then, compute the element stiffness matrix on $\hat{T}$ for this basis,

$$\hat{K} = \left( \int_{\hat{T}} \hat{\varphi}_j'(\xi)\ \hat{\varphi}_i'(\xi)\ d\xi \right)_{i,j=0}^{2}.$$

---

**Programming.**
In $C^{++}$ (or C) only! (no Fortran, no Java, no matlab)
You will need a C/$C^{++}$ compiler and an editor, or an integrated development environment (like Dev$C^{++}$, eclipse, Visual Studio, ...).

You might want to define the following data types:

```
typedef double Vec2[2];
typedef double Mat22[2][2];
```

define a vector type `Vec2` in $\mathbb{R}^2$ and a $2 \times 2$ matrix type `Mat22`, and

```
typedef double (*RealFunction)(double x);
```

defines a function type `RealFunction`.
*For storing bigger vectors, you are recommended to use the* ***vector class*** *provided on the* ***homepage****.*

**18** Design a data type `Mesh` to store the mesh information that you need later on to assemble the stiffness matrix. Make sure that your data type allows

- initializing (e.g. with an equidistant mesh with a certain number of nodes)
- asking for the number of nodes
- asking for the "coordinate" of an arbitrary node

Implement a method `void TestMesh()` which creates a mesh of $[0,1]$, performs different queries on it and prints the result to the screen.

*Hint:* Use `class` in $C^{++}$or `struct` in C.

**19** Design an *efficient* data type `SMatrix` to store the stiffness matrix $K_h$ later on, which exploits the fact that $K_h$ is tridiagonal. Make sure that your data type allows

- initializing (with a certain number of rows=columns and zero entries)
- asking for any entry in the diagonal and the two off-diagonals
- adding a value to a certain entry

Implement a method `void TestSMatrix()` which creates a tridiagonal matrix, performs different queries on it and prints the results to the screen.

*Hint:* Use `class` in C++ or `struct` in C.

**20** Write a function

```
void ElementStiffnessMatrix (double xa, double xb, Mat22& elMat);
```

which for given nodes `xa`$=x_{k-1}$ and `xb`$=x_k$ returns the element stiffness matrix `elMat`$=K_h^{(k)}$ of the element $T_k$, i.e.

$$K_h^{(k)} = \begin{pmatrix} \int_{T_k} (\varphi'_{k-1}(x))^2 dx & \int_{T_k} \varphi'_{k-1}(x)\varphi'_k(x) dx \\ \int_{T_k} \varphi'_k(x)\varphi'_{k-1}(x) dx & \int_{T_k} (\varphi'_k(x))^2 dx \end{pmatrix} = \frac{1}{h_k} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

Implement a method `void TestElementMatrix()` which creates an element stiffness matrix and prints the result to the screen.

**21** Write a function

```
void ElementLoadVector (RealFunction f, double xa, double xb,
                        Vec2& elVec);
```

which for a given function `f`$=f \in C[0, 1]$ and the nodes `xa`$=x_{k-1}$ and `xb`$=x_k$ returns the approximated 2-dimensional element load vector `elVec`$\approx f_h^{(k)}$ on the element $T_k$,

$$f_h^{(k)} = \begin{pmatrix} \int_{T_k} f(x)\varphi_{k-1}(x) dx \\ \int_{T_k} f(x)\varphi_k(x) dx \end{pmatrix}.$$

Use the trapezoidal rule to approximate the involved integrals (see lecture).

Implement a method `void TestElementVector()` which creates an element load vector for the function $f(x) = 3x + 1$ and prints the result to the screen.