C++ code 3.23.4: Function `getNeumannData()`

```
lf::mesh::utils::CodimMeshDataSet<double> getNeumannData(
    std::shared_ptr<const lf::uscalfe::FeSpaceLagrangeO1<double>> fe_space,
    const Eigen::VectorXd &mu) {
  lf::mesh::utils::CodimMeshDataSet<double> edge_vals(fe_space->Mesh(), 1, 0.0);
  // Obtain flags indicating edges on the boundary
  lf::mesh::utils::CodimMeshDataSet<bool> bded_flags{
      lf::mesh::utils::flagEntitiesOnBoundary(fe_space->Mesh(), 1)};
  // Compute the piecewise constant gradients for all cells of the mesh
  const lf::fe::MeshFunctionGradFE mf_grad_fefun(fe_space, mu);
  // Reference coordinates of barycenter of mesh
  Eigen::MatrixXd refc_center = Eigen::Vector2d(1.0 / 3.0, 1.0 / 3.0);
  // Loop over all cells of the mesh
  for (const lf::mesh::Entity *cell : fe_space->Mesh()->Entities(0)) {
    LF_VERIFY_MSG(cell->RefEl() == lf::base::RefEl::kTria(),
                  "Only triangular cells admitted!");
    // Compute constant value of the gradient for the current cell
    const Eigen::Vector2d grad_fefun = mf_grad_fefun(*cell, refc_center)[0];
    // Visit edges of the cell and check whether they are located on the
    // boundary
    std::span<const lf::mesh::Entity *const> edges{cell->SubEntities(1)};
    int ed_cnt = 0;
    for (const lf::mesh::Entity *edge : edges) {
      if (bded_flags(*edge)) {
        // Edge is located on the boundary:
        // Obtain exterior unit normals for the triangle, pick that belonging to
        // the current edge and compute its Euclidean inner product with the
        // local gradient vector.
        const Eigen::Matrix<double, 2, 4> unit_normals =
            exteriorUnitNormals(*(cell->Geometry()));
        edge_vals(*edge) = unit_normals.col(ed_cnt).dot(grad_fefun);
      }
      ed_cnt++;
    }
  }
  return edge_vals;
}
```