Concepts：对某种类型type施加限制，但不要求其属于某个特定的继承层次结构。在deal.II中通过把concept名称作为模板的typename，来给类型施加限制。比如为了限制Vector类，构造模板类：

template <typename Number>

class Vector;

这个类Vector依赖于一个作为基础的场field的类型。在这里创建的Vector可以用来存储类型为Number的对象，但同时它还定义了在向量空间上的操作如向量加减、求向量范数等（从而区别于std::vector）。因而Number类必须满足一些条件（满足某种concept）：必须代表了数学上可以称为“场”的东西（即可以加、乘、除、求绝对值等）。因而所谓“concept”其关键就在于一个type必须满足什么条件才能成为合格的模板参数。

本页介绍了deal.II中用到的一些concepts。

**DoFHandlerType**

Deal.II中包括了DoFHandlerH和hp::DoFHandler用于管理网格上的自由度。这两者虽然没有任何继承关系，但非常相似。

**MatrixType**

如果某个类模板需要以MatrixType做模板参数，那么这个类应该实现四个成员函数（线性变换的函数如矩阵乘向量）。LinearOperator使这种工作简化了很多。

可以把MatrixType设想成下面这样的一种基类（实际上SparseMatrix的接口基本上就是这样的）：

Class MatrixType{

Public：

template<typename VectorType>

virtual void vmult(VectorType& u, const VectorType& v) const = 0;

template <typename VectorType>

virtual void Tvmult(VectorType &u, const VectorType &v) const =0;

template <typename VectorType>

virtual void vmult\_add(VectorType &u, const VectorType &v) const =0;

template <typename VectorType>

virtual void Tvmult\_add(VectorType &u, const VectorType &v) const =0;

};

C++中的模板函数不能是虚函数（所以deal.II中并没有使用这种方法），但这仍然是种帮助理解的方式。可以使用PointerMatrixAux类替代手工来实现vmult\_add和Tvmult\_add。

**MeshType**

可以把网格看做一个cells的集合。C++中集合称为容器containers，其特点是可以在元素上进行迭代（循环）。MeshType概念就是任何定义了网格操作的容器。Triangulation，DoFHandler，hp::DoFHandler都可看做是网格容器。这些类的公共接口的最重要的特征就是允许获得指向某个元素的迭代器（指针）。因为这些接口是通用的，所以不管是Triangulation还是DoFHandler，它们可以使用同一套函数（在GridTools命名空间下）。

另一方面，网格不同于一般的容器，它是具有某种层次结构的。用户可在只在某一层级上进行循环。

函数接收一个模板参数，像这样：

template <template <int, int> class MeshType>或

template <typename MeshType>

MeshType类可以概括为具有如下特征的类：

1）一个名为active\_cell\_iterator的typedef

2）一个方法：get\_triangulation()，返回对底层网格几何描述的引用

3）一个方法：begin\_active()，返回指向第一个激活网格的指针

4）一个静态成员值：dimension，表示对象存在于几维空间

5）一个静态成员值：space\_dimension，表示对象的维度（如存在于3D的2D表面有space\_dimension=2）

**Number**

表述标量，通常是一个场元素的有限精度近似。最常见的如：double和float，deal.II也支持std::complex<T>

**PolynomialType**

详见Polynomials and polynomial spaces中的描述。简单讲，像下面这种接口的就叫多项式：

template <int dim>

class PolynomialType

{

virtual void compute (const Point<dim> &unit\_point,

std::vector<Tensor<1,dim> > &values,

std::vector<Tensor<2,dim> > &grads,

std::vector<Tensor<3,dim> > &grad\_grads) const =0;

}

**PreconditionerType**

基本上是MatrixType的同义词，但一般只定义了vmult()和Tvmult()函数。在这里应该把vmult()当做对向量施加某种线性变换的逆的近似，而非对向量的线性变换。

**RelaxationType**

在多块网格方法中能够驰豫relaxation的对象。同时满足下列接口及MatrixType类要求的即RelaxationType：

class RelaxationType

{

public:

template <typename VectorType>

virtual void step(VectorType &u, const VectorType &v) const =0;

template <typename VectorType>

virtual void Tstep(VectorType &u, const VectorType &v) const =0;

};

Where these two member functions perform one step (or the transpose of such a step) of the smoothing scheme.

这些函数完成的操作是：

C:\Users\zeng\AppData\Local\Temp\1510575025(1).png

**SparsityPatternType**

几乎所有能以稀疏模式作参数的函数都能以SparstyPattern或DynamicSparsityPattern作参数。详见Sparsity Pattern模块。

**StreamType**

在C++得到新的stream类很困难。为了绕过这一困难，有些函数定义了<<操作，使得输出任何类型的output stream变得容易。

**VectorType**

Deal.II支持很多不同的vector类，包括绑定其他库的vector。一些例子如：Vector，TrilinosWrappers::MPI::Vector和BolckVector。