**ParameterHandler**

简介(从step 19摘过来的)：

为了从外部得到程序需要的参数，我们通过以下三步来实现：

1. 通过ParameterHandler对象prm来声明可能需要的全部变量：prm.declare\_entry()

2. 由ParameterHandler类根据这些参数来读入某个参数文件：prm.parse\_input()

3. 程序通过prm得到参数的值：prm.get()

prm其实起到一个中介的作用，它对外可方便地控制不同类型的参数输入，对内可以把参数正确地传递给内部变量。下面分别介绍这三个函数：

首先，ParameterHandler::declare\_entry()函数，用于声明在input文件的当前部分存在哪些参数名称（可以使用函数ParameterHandler::enter\_subsection, leave\_subsection进入或离开文件的subsection，类似cd dir和cd ..命令）。当声明一个参数时，必须指明它的名称和默认值，以防参数之后没在文件中被列出。此外，这个函数有额外的可选arguments指明了某个参数需要满足的pattern，比如它必须是整数，以及可能在之后用于解释该参数含义的帮助文本。

当所有参数都被声明了之后，就可以使用ParameterHandler::parse\_input()函数来读入参数了。这个函数有很多变体，可以从文件流中读入，可以从文件名读入，也可以使用一个string并parse它。当读入参数的时候，这个类会确保只有之前声明过的参数被列在了input文件中，且它们的值的类型满足指定的pattern。如果input使用了未被声明的变量或不满足类型的变量值，会拒绝读入并抛出异常。

最后，用户的应用程序可以遍历参数树的subsection，并使用ParameterHandler::get函数来获取声明过的参数的值。ParameterHandler::get仅仅把一个参数作为string来返回，而ParameterHandler::get\_integer，ParameterHandler::get\_double等则进一步把数据转换为指定的类型。

因而，使用ParameterHandler提供了一种相当灵活的机制用于处理各种较复杂的输入文件。

这个类提供了用于输入文件的标准接口。输入可以是在文件中或内存的streams或strings中通过text的方式给出：

set Time step size = 0.3

set Geometry =[0,1]x[0,3]

输入可能按subsection trees来组织，以便提供一种逻辑结构。

**Declaring entries**

为了利用好ParameterHandler对象，我们必须先明确输入文件中可能或不可能包含哪些条目。按下列方式实现：

…

parameterHandler prm;

prm.declare\_entry(“Time step size”, “0.2”, Patterns::Double(), “Some documentation”);

prm.declare\_entry(“Geometry”,”[0,1]x[0,1]”,Patterns::Anything());

…

每个条目都通过函数declare\_entry()进行声明。第一个参数是条目的名称。第二个参数是默认值。第三个参数是一个正则表达式，输入(及默认值)必须匹配它。在Patterns中定义了一些这样的正则表达式。这个参数可以省略，则默认成为Patterns::Anything，即匹配任何输出string的pattern。第四个参数可用于记录某条条目的内容或期望的格式，当使用print\_parameters()函数写出所有条目时，它的值会以注释的方式被写出，以助于更好地理解某个参数文件，它也可以被省略。

条目可能处于subsections中，从而形成一种输入树。例如，用于线性求解器的输出参数可以分类到一个名为Linear solver的subsection中去，如下：

…

LinEq eq;

eq.declare\_parameters (prm);

…

void LinEq::declare\_parameters (ParameterHandler &prm) {

prm.enter\_subsection("Linear solver");

{ prm.declare\_entry ("Solver",

"CG",

Patterns::Selection("CG|GMRES|GaussElim"),

"Name of a linear solver for the inner iteration");

prm.declare\_entry ("Maximum number of iterations",

"20",

ParameterHandler::RegularExpressions::Integer());

... }

prm.leave\_subsection ();

}

Subsections可以是嵌套的。例如，一个非线性求解器可能有一个线性求解器作为成员对象。那么函数调用树可能类似于（如果类NonLinEq有一个类型为LinEq的成员变量eq）：

void NonLinEq::declare\_parameters (ParameterHandler &prm) {

prm.enter\_subsection ("Nonlinear solver");

{ prm.declare\_entry ("Nonlinear method",

"Newton-Raphson",

ParameterHandler::RegularExpressions::Anything());

eq.declare\_parameters (prm);

}

prm.leave\_subsection ();

}

//省略了一段//

**Input files and special characters**

对于上面的第一个例子，输入文件可能如下：

...

subsection Nonlinear solver

set Nonlinear method = Gradient

# this is a comment

subsection Linear solver

set Solver = CG

set Maximum number of iterations = 30

end

end

... # other stuff

在条目名称中不允许使用下列符号：#，{，}，|.

**Including other input files**

一个输入文件可以使用如下语法来包含其他文件：

...

include some\_other\_file.prm

...

这样引用的文件会在当前目录下被搜索。

**Reading data from input sources**

为了读入input有三种方法：从std::istream对象读，从一个给定名称的文件读，从内存的string中读。

ParameterHandler prm;

...

// declaration of entries

...

prm.parse\_input (std::cin); // read input from standard in,

// or

prm.parse\_input ("simulation.prm");

// or

char \*in = "set Time step size = 0.3 \n ...";

prm.parse\_input\_from\_string (in);

...

**Using the ParameterHandler Graphical User Interface**

//略

**Worked Example**

#include <deal.II/base/parameter\_handler.h>

#include <iostream>

#include <string>

using namespace dealii;

class LinearEquation

{

public:

static void declare\_parameters(ParameterHandler& prm);

void get\_parameters(ParameterHandler& prm);

private:

std::string method;

int max\_iterations;

};

class Problem

{

private: **//Problem类的如下参数成员需要介由prm接口从外部获取。**

LinearEquation eq1,eq2;

std::string matrix1, matrix2;

std::string outfile;

public:

static void declare\_parameters(ParameterHandler& prm);

void get\_parameters(ParameterHandler& prm);

void do\_something();

};

void LinearEquation::declare\_parameters(ParameterHandler& prm) //声明参数

{

//declare parameters for the linear solver in a subsection

prm.enter\_subsection(“Linear solver“);

{

prm.declare\_entry(“Sovler“,

“CG“,

Patterns::Selection(“CG|BiCGStab|GMRES“),

“Name of a linear solver for the inner iteration“);

prm.declare\_entry(“Maximum number of iterations“,

“20“,

}

prm.leave\_subsection();

}

void LinearEquation::get\_parameters(ParameterHandler& prm) //获取参数

{

prm.enter\_subsection(“Linear solver“);

{

method = prm.get(“Solver“);

max\_iterations = prm.get\_integer(“Maximum number of iterations“);

}

prm.leave\_subsection();

std::cout<<“LinearEquation: method=“<<method<<“,,max\_iterations=“<<max\_iterations<<std::endl;

}

void Problem::declare\_parameters (ParameterHandler &prm){ //声明参数

// first some global parameter entries

prm.declare\_entry ("Output file",

"out",

Patterns::Anything(),

"Name of the output file, either relative to the present path or absolute");

prm.declare\_entry ("Equation 1",

"Laplace",

Patterns::Anything(),

"String identifying the equation we want to solve");

prm.declare\_entry ("Equation 2",

"Elasticity",

Patterns::Anything());

// declare parameters for the first equation

prm.enter\_subsection ("Equation 1 Settings");

{

prm.declare\_entry ("Matrix type",

"Sparse",

Patterns::Selection("Full|Sparse|Diagonal"),

"Type of the matrix to be used, either full, "

"sparse, or diagonal");

LinearEquation::declare\_parameters (prm); // for eq1

}

prm.leave\_subsection ();

// declare parameters for the second equation

prm.enter\_subsection ("Equation 2 Settings");

{

prm.declare\_entry ("Matrix type",

"Sparse",

Patterns::Selection("Full|Sparse|Diagonal"));

LinearEquation::declare\_parameters (prm); // for eq2 }

prm.leave\_subsection ();

}

void Problem::get\_parameters (ParameterHandler &prm) //获取参数

{

// entries of the problem class

outfile = prm.get ("Output file");

std::string equation1 = prm.get ("Equation 1"),

equation2 = prm.get ("Equation 2");

// get parameters for the first equation

prm.enter\_subsection ("Equation 1 Settings");

{

matrix1 = prm.get ("Matrix type");

eq1.get\_parameters (prm); // for eq1

}

prm.leave\_subsection ();

// get parameters for the second equation

prm.enter\_subsection ("Equation 2 Settings");

{

matrix2 = prm.get ("Matrix type");

eq2.get\_parameters (prm); // for eq2

}

prm.leave\_subsection ();

std::cout << " Problem: outfile=" << outfile << '\n'

<< " eq1=" << equation1 << ", eq2=" << equation2 << '\n'

<< " matrix1=" << matrix1 << ", matrix2=" << matrix2

<< std::endl;

}

void Problem::do\_something ()

{

// While this example does nothing here, at this point in the program

// all of the parameters are known so we can start doing computations.

}

int main ()

{

ParameterHandler prm;

Problem p;

**p.declare\_parameters (prm);** //**在这个函数下调用prm.declare\_entry()来声明在input文件中可能存在哪些参数条目。此函数对p无影响，只是改变了prm。**

// read input from "prmtest.prm"; giving argv[1] would also be a good idea

**prm.parse\_input ("prmtest.prm");** **//扫描外部文件每一行，解析参数值，此时prm知道了所有参数的名称及对应值。但p仍未改变。**

// print parameters to std::cout as ASCII text

std::cout << "\n\n"; prm.print\_parameters (std::cout, ParameterHandler::Text);

// get parameters into the program

std::cout << "\n\n" << "Getting parameters:" << std::endl;

**p.get\_parameters (prm);** /**/利用prm给p中的成员赋值。此时p中的参数成员赋值完毕。**

// now run the program with these input parameters

p.do\_something ();

}

下面是输入文件（名为“prmtest.prm“）：

# first declare the types of equations

set Equation 1 = Poisson

set Equation 2 = Stokes

subsection Equation 1 Settings

set Matrix type = Sparse

subsection Linear solver # parameters for linear solver 1

set Solver = Gauss-Seidel

set Maximum number of iterations = 40

end

end

subsection Equation 2 Settings

set Matrix type = Full

subsection Linear solver

set Solver = CG

set Maximum number of iterations = 100

end

end

以下是程序的输出：

Line <8> of file <prmtest.prm>:

The entry value

Gauss-Seidel

for the entry named

Solver

does not match the given pattern

[Selection CG|BiCGStab|GMRES ]

# Listing of Parameters

# ---------------------

# String identifying the equation we want to solve

set Equation 1 = Poisson # default: Laplace

set Equation 2 = Stokes # default: Elasticity

# Name of the output file, either relative to the present path or absolute

set Output file = out

subsection Equation 1 Settings

# Type of the matrix to be used, either full, sparse, or diagonal

set Matrix type = Sparse

subsection Linear solver

set Maximum number of iterations = 40 # default: 20

# Name of a linear solver for the inner iteration

set Solver = CG

end

end

subsection Equation 2 Settings

set Matrix type = Full # default: Sparse

subsection Linear solver

set Maximum number of iterations = 100 # default: 20

# Name of a linear solver for the inner iteration

set Solver = CG

end

end

Getting parameters:

LinearEquation: method=CG, max\_iterations=40

LinearEquation: method=CG, max\_iterations=100

Problem: outfile=out

eq1=Poisson, eq2=Stokes

matrix1=Sparse, matrix2=Full