C++, オブジェクト指向プログラミングの基礎

OpenFOAM®ソースコードの構造とコンパイル

2019年3月29, 30日
オープンCAE講習会@富山

富山県立大学　中川慎二

目次

- OpenFOAMとは
- OpenFOAMとC++
- C++の基礎の基礎　ハンズオン

OpenFOAMとは

- オブジェクト指向の考え方で設計され，C++プログラミング言語により実装された，CFDに必要な機能の集合体である。
- OpenFOAMとは、CFDに必要な機能を追加した拡張版C++言語と考えることもできる。
- OpenFOAMのプログラム（ソースコード）を深く理解するためには，C++言語，オブジェクト指向プログラミング，ジェネリックプログラミングなどの考え方を理解していることが必要となる。すべてを理解することは，非常にむずかしい。
- すべてを理解していなくても，各クラス・関数等の使い方を知っていれば，ソースコードを理解したり改造したりすることができる。

OpenFOAMとC++

OpenFOAMのソースコード

- C++ 言語
- オブジェクト指向プログラミング
  - カプセル化 (振る舞いの隠蔽とデータ隠蔽)
  - インヘリタンス (継承) -- クラスベースの言語
  - ポリモーフィズム (多態性、多相性) -- 型付きの言語
    - オーバーロード（多重定義） 同じ名前で引数の異なる関数
    - オーバーライド　親クラスの関数を子クラスで上書き
- ジェネリックプログラミング
  - データ型に依存しないコード。Templateを利用。
- OpenFOAM とは、C++言語でCFDを実行するための工具一式(toolkit)である。

C++ クラスとは

- 部品（オブジェクト）の設計図
- 様々な値（状態，属性）と，それを操作するための機能（function,メソッド，関数）を含む
- クラスは、値と関数の集まりである
- この設計図（クラス）に基づいて，プログラム実行時に，部品（オブジェクト）が作られる
- オブジェクト生成時にはコンストラクタが働く

一般コード例

```
int    n(7);
int    i = 10;
```

型名　変数名（初期値）

OpenFOAMコード例

```
dimensionedScalar    DT( x, y);
```

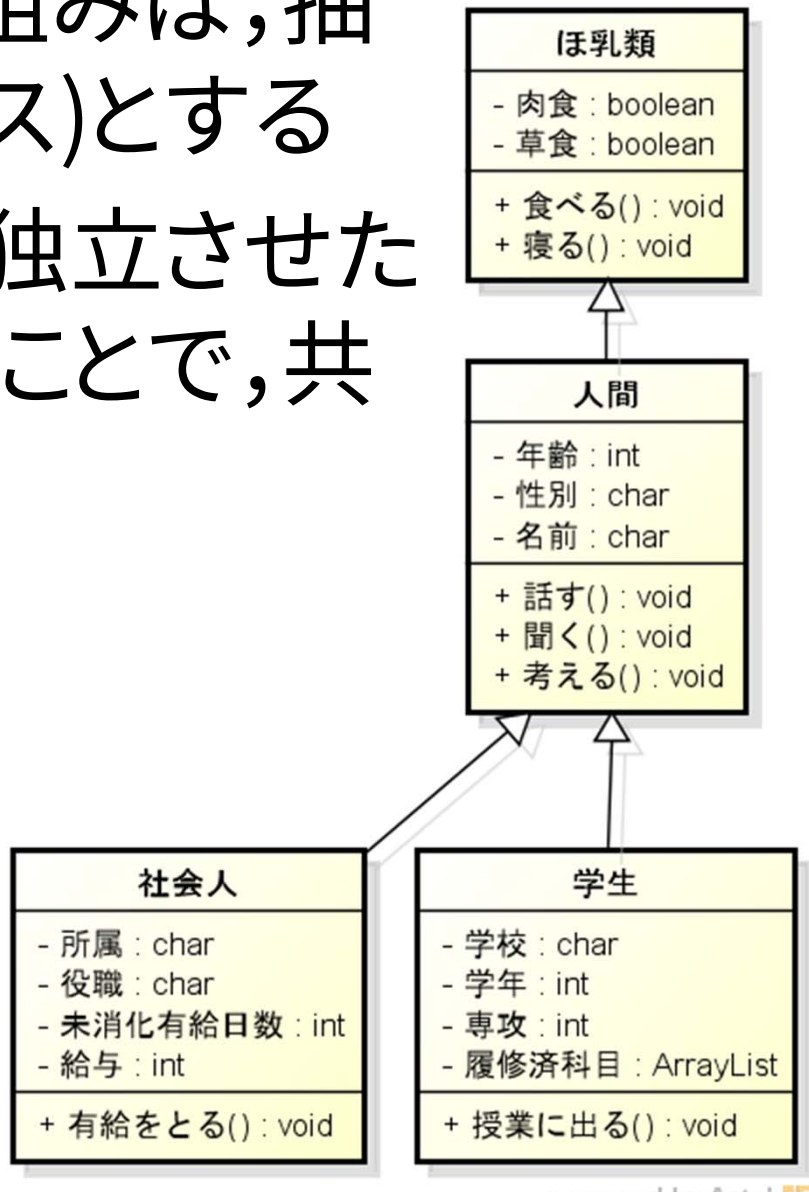
クラス名　　オブジェクト名（初期値）

C++ クラスの継承

- 複数のクラスに共通する仕組みは，抽象化した独立クラス(親クラス)とする
- 複数のクラス(子クラス)が，独立させたクラス(親クラス)を継承することで，共通の機能を実現する

例

- 親クラス＝人間
- 子クラス＝学生，社会人



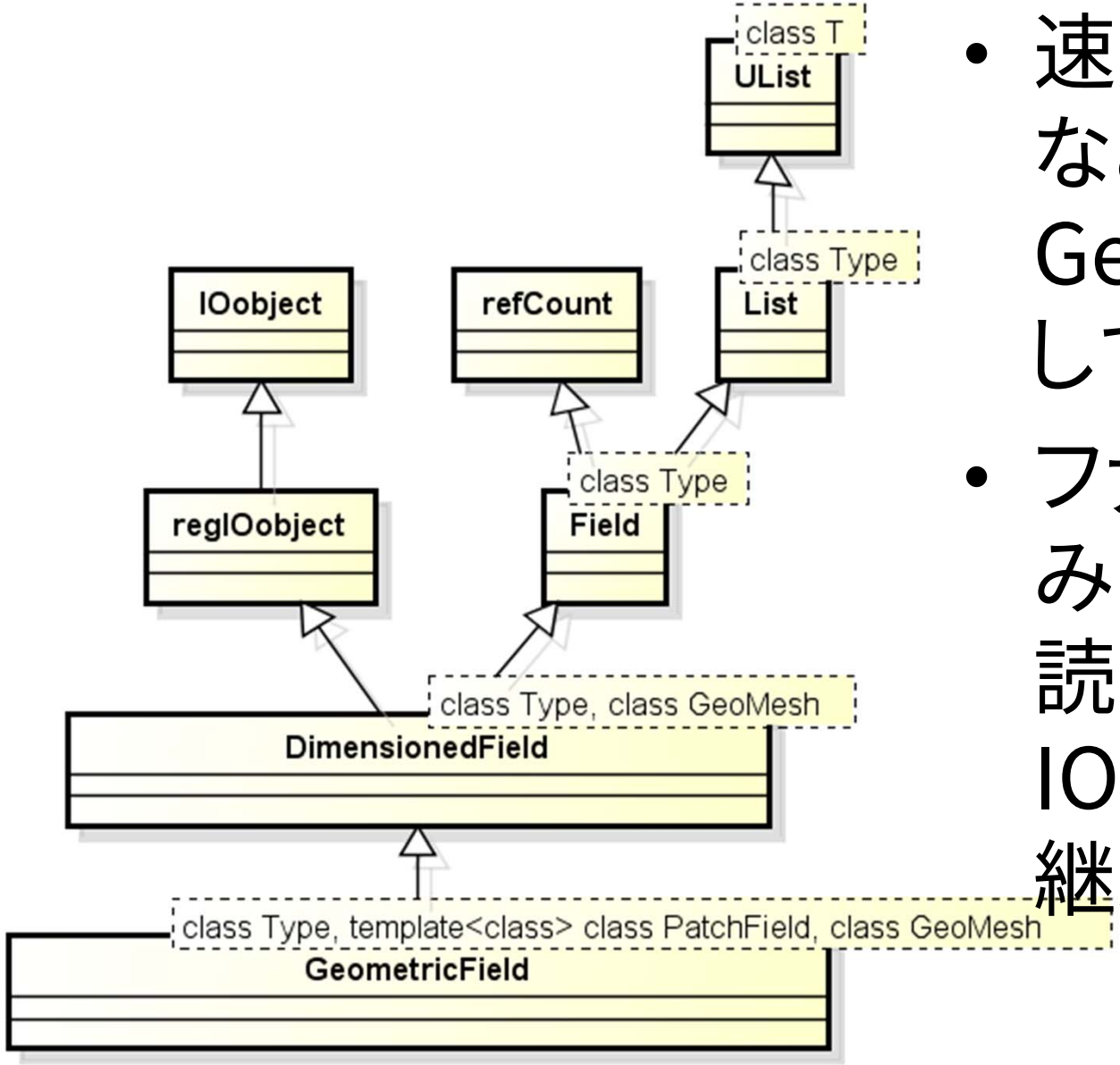
よく使う クラス

	非フィールド値 場所によらず一定	フィールド値 場所によって値が変わる	
		セル面での値	セル体積（中心）での値
スカラー	dimensionedScalar dimensioned<scalar>	surfaceScalarField GeometricField<scalar, fvsPatchField, surfaceMesh>	volScalarField GeometricField<scalar, fvPatchField, volMesh>
ベクトル	dimensionedVector dimensioned<vector>	surfaceVectorField GeometricField<vector, fvsPatchField, surfaceMesh>	volVectorField GeometricField<vector, fvPatchField, volMesh>

上の行は，typedefで定義された別名
下の行が本来の定義

templateクラスで多様なタイプに対応

GeometricFieldクラスの継承関係



- 速度U，圧力p，温度Tなどのデータは，GeometricField型として保存されている。
- ファイルへの書き込み／ファイルからの読み込みなどは，IOobjectクラスから継承している。

ソルバ

- 特定の問題を解くために，OpenFOAMのコードを組み合わせたプログラム。
- ソルバのソースコードは，とてもシンプル。難しい作業は，部品（ライブラリ，クラス）に任せる。
- まとまった作業は別ファイルに記述し，includeすることで，読みやすさを保つ。
- 様々な部品（オブジェクト）が，協調しながら，目的を果たす。
- 部品どうしは，適切な独立性を持っている。
- ソルバのソースコードは，ソルバ名.Cとなっている。

ライブラリ

- 特定の機能を実現するのに必要な部品を集めたもの。
- 関連する複数のクラスから，1つのライブラリを作成する。
- 例えば，incompressibleTurbulenceModelsライブラリの中に，kEpsilonクラスや，Smagorinskyクラス etc. が含まれている。
- srcディレクトリ内で，Makeディレクトリが存在するディレクトリから，ライブラリが作られる。

チュートリアル

- ソルバ，ライブラリ，クラスなどの機能・使い方を説明するために用意された例題。
- チュートリアルは，説明書の一部と考えられる。
- チュートリアルを実行しながら，理解を深めることが大切。
- ソースコード改造時には，その改造内容に応じたチュートリアルを作成する。

ハンズオン

https://github.com/snaka-dev/Training_Cpp_Introduction

http://eddy.pu-toyama.ac.jp/bbtb56fof-126/#_126

solve (A-B); なのか，
solve (A==B); なのか？

質問

laplacianFoam.C では，

```
        solve
        (
            fvm::ddt(T) - fvm::laplacian(DT, T)
        );
```

と表記してありますが、

```
        solve
        (
            fvm::ddt(T) == fvm::laplacian(DT, T)
        );
```

としても良いのですか？

回答

上記2つの表記は，どちらでも良いです。同じ意味になります。

　これには，演算子”==”のオーバーロードが関係しています。

　ソースコードを見ながら確認します。

　演算子は，どのクラスに対しての演算子なのか，に注意して調べる必要があります。

　今の質問では，fvm::ddt(T) と fvm::laplacian(DT, T) との演算です。両者のクラスは何になるでしょう?

fvmDdt.Cを見る。
fvm::ddt(T) の戻り値＝ tmp<fvMatrix<Type> > 型
このTypeは，laplacianメソッドに渡したＴと同じ。

```
43  template<class Type>
44  tmp<fvMatrix<Type>>
45  ddt
46  (
47      const GeometricField<Type, fvPatchField, volMesh>& vf
48  )
```

　ということは，fvm::ddt(T) == fvm::laplacian(DT, T) では，tmp<fvMatrix<Type> >に対しての演算子”==”を確認する必要があります。

fvmLaplacian.Cを見る。
fvm::laplacian(DT, T) の戻り値＝ tmp<fvMatrix<Type> > 型
このTypeは，laplacianメソッドに渡したＴと同じ。

```
179   template<class Type, class GType>
180   tmp<fvMatrix<Type>>
181   laplacian
182   (
183       const dimensioned<GType>& gamma,
184       const GeometricField<Type, fvPatchField, volMesh>& vf
185   )
```

　ということは，fvm::ddt(T) == fvm::laplacian(DT, T) では，tmp<fvMatrix<Type> >に対しての演算子”==”を確認する必要があります。

fvMatrix.H にGlobal operators として operator== が記載されています。

```
617 template<class Type>
618 tmp<fvMatrix<Type> > operator==
619 (
620     const tmp<fvMatrix<Type> >&,
621     const tmp<fvMatrix<Type> >&
622 );
```

fvMatrix.C に定義がある。

演算子“==”の左側 tmp<fvMatrix<Type> > をtAという名前で，右側 tmp<fvMatrix<Type> > をtBという名前で受取る。

演算結果として，return の後にあるもの（tA - tB）が戻される。

つまり，左辺ー右辺の行列（tmp<fvMatrix<Type> > 型）を戻すことになる。

```
1446 template<class Type>
1447 Foam::tmp<Foam::fvMatrix<Type> > Foam::operator==
1448 (
1449     const tmp<fvMatrix<Type>>& tA,
1450     const tmp<fvMatrix<Type>>& tB
1551 )
1552 {
1553     checkMethod(tA(), tB(), "==");
1554     return (tA - tB);
1555 }
```

なお，ここで使った演算子 ”- ” 自体も，fvMatrixクラス用にオーバーロードしたものが使われています。オリジナルのコードで使われているのと同じです。こちらは直感的にわかるので，疑問に感じないかもしれません。

（参考）
オペレータ(演算子)のオーバーロードについて。
http://homepage2.nifty.com/well/Operator.html
http://www.geocities.jp/ky_webid/cpp/language/017.html
http://www.geocities.jp/penguinitis2002/study/OpenFOAM/OpenFOAM-cpp_primer.html