

# OpenFoamのためのC/C++

第5回 IDEによるデバッグ・ソースコード管理 (\*)

田中昭雄

(\*) 分量の問題で割愛

# 目的

この勉強会の資料があれば、  
OpenFoamカスタマイズ時にC/C++で迷わない

# 予定

- 第1回 メモリ管理
- 第2回 CFDの例で勉強するクラス
- 第3回 OpenFOAMで勉強するテンプレート
- 第4回 ソルバーカスタマイズの基本
- 第5回 IDEによるデバッグ・~~ソースコード管理~~
- 第6回 未定(できればUtilitiesを使った何か)

# 今回のテーマ

IDEによるデバッグ・ソースコード管理

## 利用環境

- Ubuntu 12.04.1 LTS
- Oracle Java 7 runtime
- Eclipse CDT Galileo SR2 64bit
- OpenFoam v2.1.1

# Agenda

- Eclipseインストール
- 使い方
- OpenFOAMカスタマイズ

# Eclipseとは

Eclipseはソフトウェア開発を効率化するためのソフトウェア

Eclipseインストール

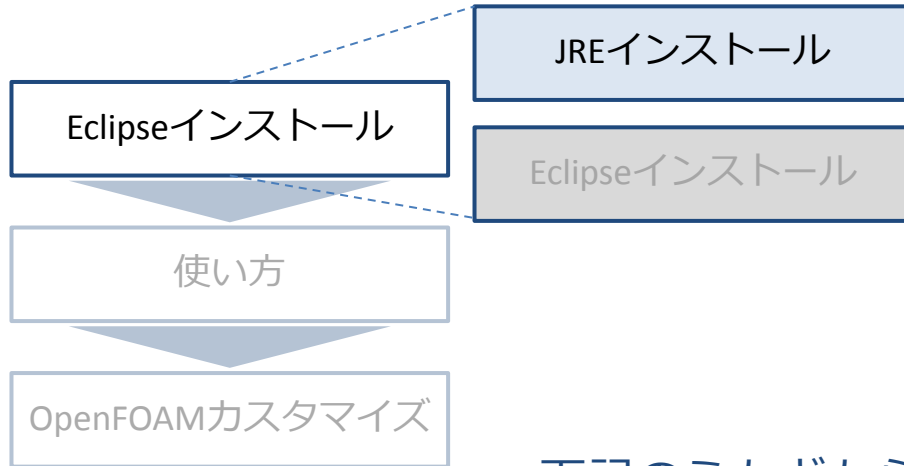
使い方

OpenFOAMカスタマイズ

- Eclipse
  - Javaで開発されているソフトウェア開発統合開発環境
  - 様々なプログラミング言語に対応するバージョンがある
  - Eclipse CDTによりC/C++を利用して開発が可能
- 統合開発環境
  - コーディング、デバッグ、ビルドを効率化するための様々な機能を持ったソフトウェア

# Eclipseインストール

## JRE (Java Runtime Environment)インストール



### JRE (Java Runtime Environment)

- Javaで開発されたソフトウェア実行に必要なソフトウェア
- Eclipseを実行するために必要
- インストールした後、明示的なJRE実行は不要

下記のうちどちらかをインストール：

### OpenJDK

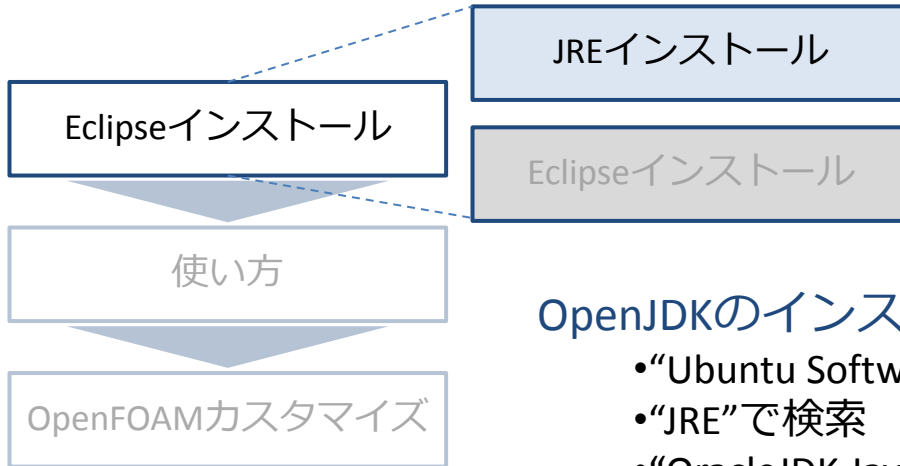
- GUIからインストール可能  
(\*apt-getを利用してもOK)

### Oracle版

- CUIからインストール  
(apt-getを利用)

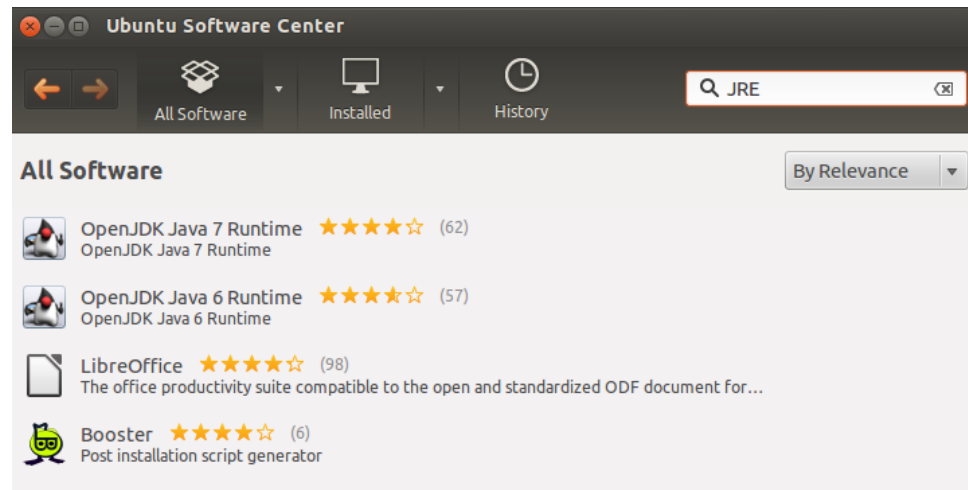
# Eclipseインストール

## JRE (Java Runtime Environment)インストール



OpenJDKのインストール（Ubuntu Software Centerを利用）：

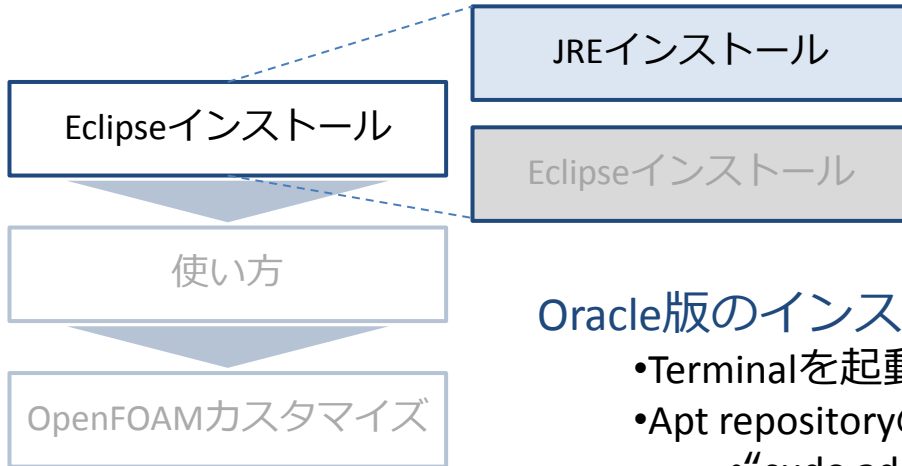
- “Ubuntu Software Center”を起動
- “JRE”で検索
- “OracleJDK Java 7 Runtime”インストール





# Eclipseインストール

## JRE (Java Runtime Environment)のインストール



### Oracle版のインストール（apt-getを利用）：

- Terminalを起動
- Apt repositoryの更新
  - “sudo add-apt-repository ppa:webupd8team/java”
  - “sudo apt-get update”
- インストール
  - “sudo apt-get install oracle-java7-installer”

# Eclipseインストール

## Eclipseのダウンロード



### •下記サイトよりダウンロード

<http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/galileosr2>



### Download Links

Windows 32-bit  
Mac OS X(Carbon)  
Mac OS X(Cocoa 32)  
Mac OS X(Cocoa 64)  
Linux 32-bit  
Linux 64-bit

### •32bit / 64bitの違いがあるので注意 (\*)

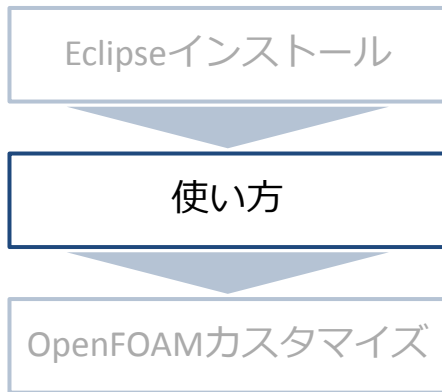
### •適当なディレクトリに解凍

•~/Eclipseディレクトリなど

(\*) Terminalで"uname -a"を実行。"x86\_64"が含まれていたら64bit / 含まれていなかったら32bit

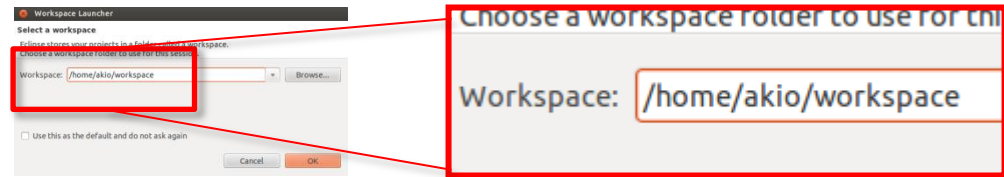
# 使い方

## 起動・ワークスペース・プロジェクト



- 起動

- ワークスペースの指定
- ホームディレクトリ以下であればどこでも良い



- ワークスペースとは

- 複数のプロジェクトをまとめたファイル置き場
- 同じワークスペースのプロジェクトは設定を共有

- プロジェクトとは

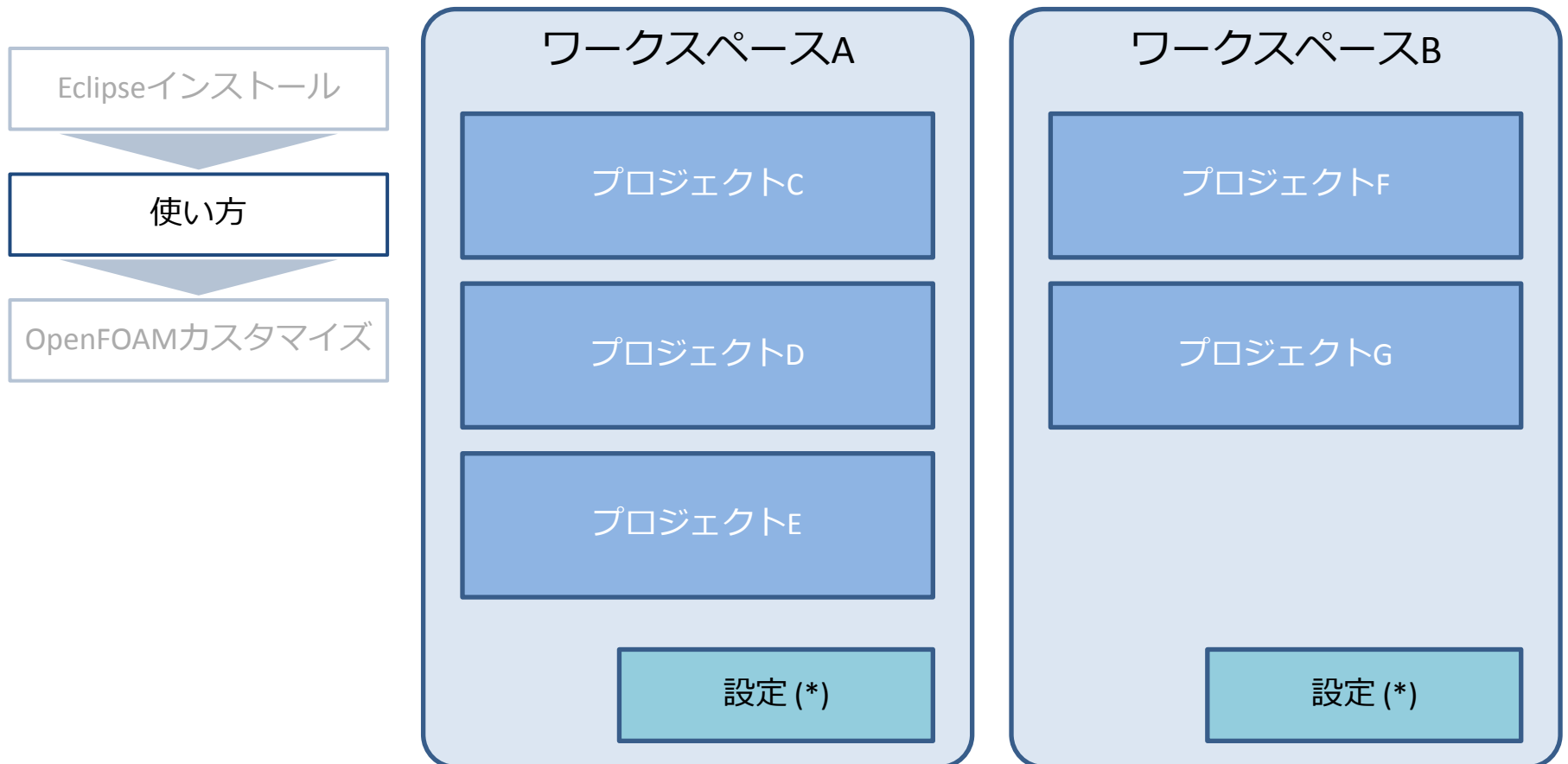
- 開発するアプリケーションを構成するまとめ
- ビルドするとアプリケーション実行形式が生成 (\*)

OpenFOAM  
カスタマイズソルバーが  
プロジェクトと1対1で  
対応するイメージ

(\*) 厳密には、1つのライブラリもしくは1つのアプリケーションがプログラムに対応

# 使い方

## ワークスペース・プロジェクトと関係



(\*) エディタ、言語設定などのみ共有。ビルド設定などはプロジェクト個別に指定される。

# 使い方

## 画面の構成

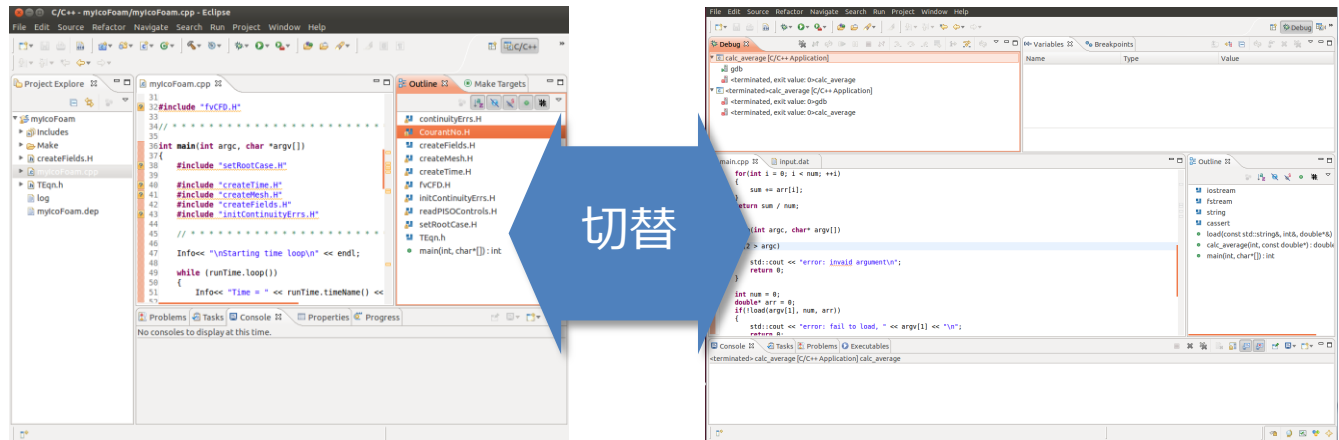
ソースコード編集用とデバッグ用の画面構成が存在

- 双方を切り替えながら効率的に開発

Eclipseインストール

使い方

OpenFOAMカスタマイズ

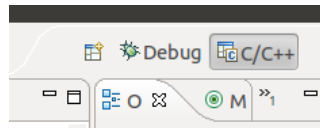


**C/C++パースペクティブ**  
ソースコード編集に特化した  
画面構成

**Debugパースペクティブ**  
デバッグ機能に特化した  
画面構成

## 切替方法

- ウィンドウ右上の「C/C++」「Debug」ボタンを利用(\*)



(\*)  
「Debug」ボタンは初めてデバッグ実行した際に  
生成されます。

# 使い方

## C/C++パースペクティブ

The screenshot shows the Eclipse IDE interface with several components highlighted by red boxes and labeled with red text:

- メニュー** (Menu): Located at the top of the window.
- ツールバー** (Toolbar): Located below the menu.
- Project Explorer**: Located on the left side, showing the project structure.
- エディタ** (Editor): The central area showing the source code of `mylcoFoam.cpp`.
- Outline**: Located on the right side, showing the project's outline and symbols.
- Problems / Console / Progress ... etc.**: Located at the bottom of the window, used for displaying build and execution results.

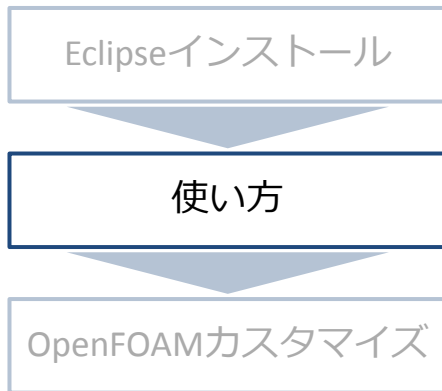
On the left side of the slide, there is a vertical flow diagram with three boxes:

- Top box: Eclipseインストール (Eclipse Install)
- Middle box: 使い方 (Usage)
- Bottom box: OpenFOAMカスタマイズ (OpenFOAM Customization)

Arrows point downwards between these boxes, indicating a sequential process.

# 使い方

## Debugパースペクティブ



メニュー

ツールバー

Debug ウィンドウ (スタックトレース)  
一時停止中の関数呼び出し状態表示

Variables / Breakpoints... etc.  
一時停止中の変数値 /  
設定中のブレークポイントリストなど

エディタ

コード編集 / 一時停止中の位置表示

Outline  
エディタ内情報

Problems / Console ... etc.  
ビルド・実行結果を表示

# 使い方

## 簡単なプログラム作成

Eclipseインストール

使い方

OpenFOAMカスタマイズ

### 数列ファイルを読み込んで平均値を標準出力

- プログラム名

  - calc\_average

- 実行方法

  - “calc\_average [数列ファイルパス]”

- 数列ファイルのフォーマット

  - 1行目：自然数値、数列の要素数

  - 2行目以降：

    - 各行に1つの実数値

    - 1行目で指定された要素数だけ行が続く

(例)

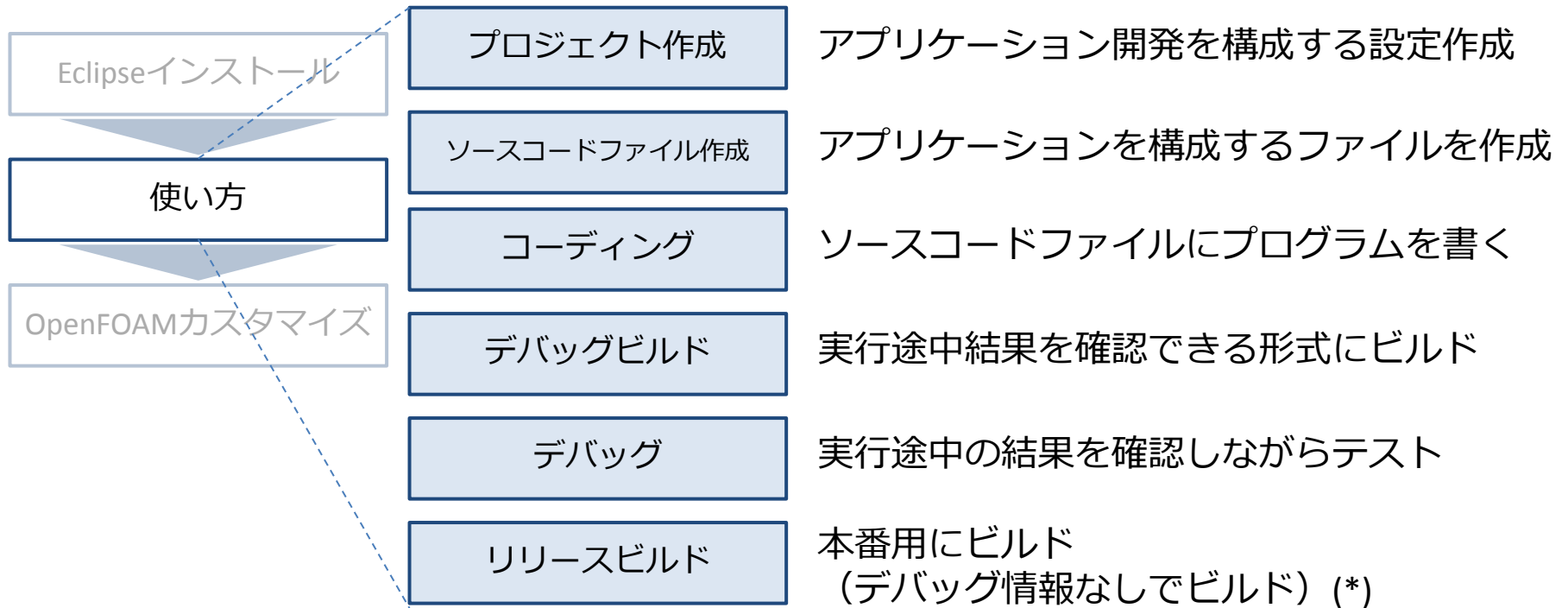
5つの要素を持つ数列ファイル

```
5
24.2
-30.5
43.9
0.8
7.1
```



# 使い方

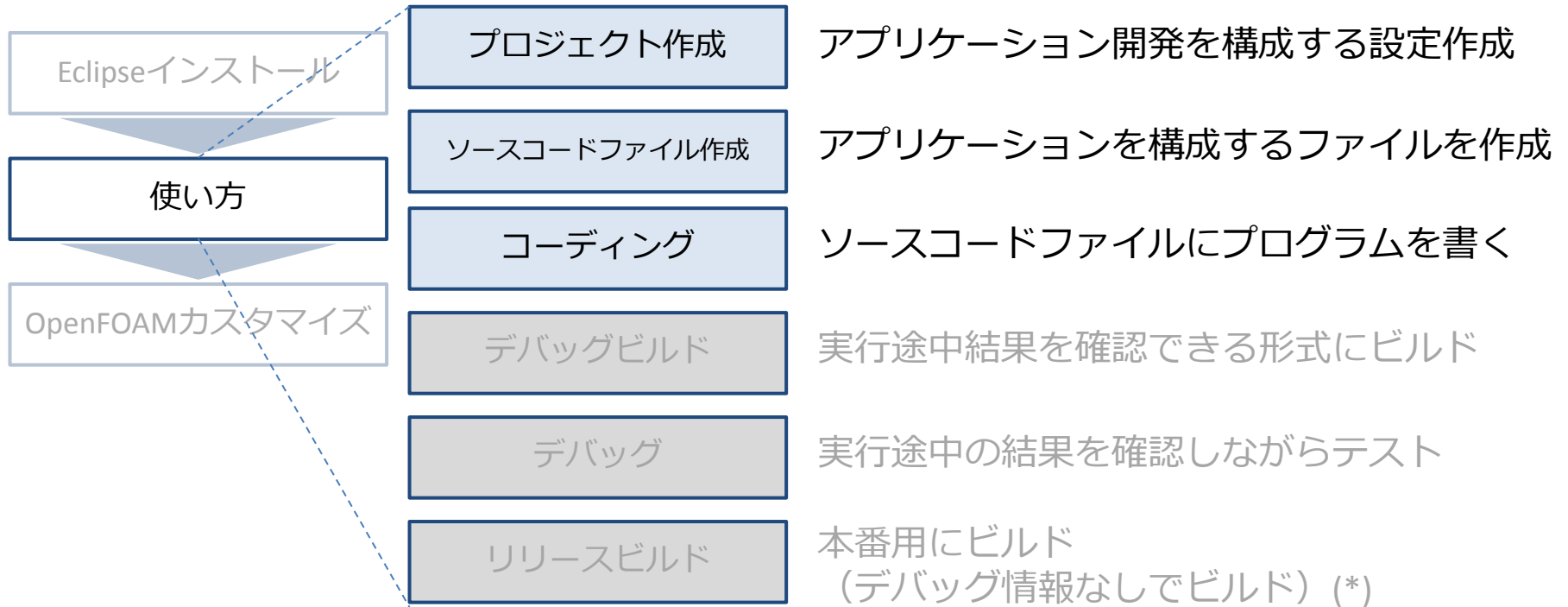
## Eclipseを使った簡単なソフトウェア開発の流れ



(\*) リリースビルドとデバッグビルドの違いは、ビルド情報生成・利用の有無、最適化オプションなど。実行速度が大幅に異なる。プロジェクト別の設定。

# 使い方

## Eclipseを使った簡単なソフトウェア開発の流れ



# 使い方

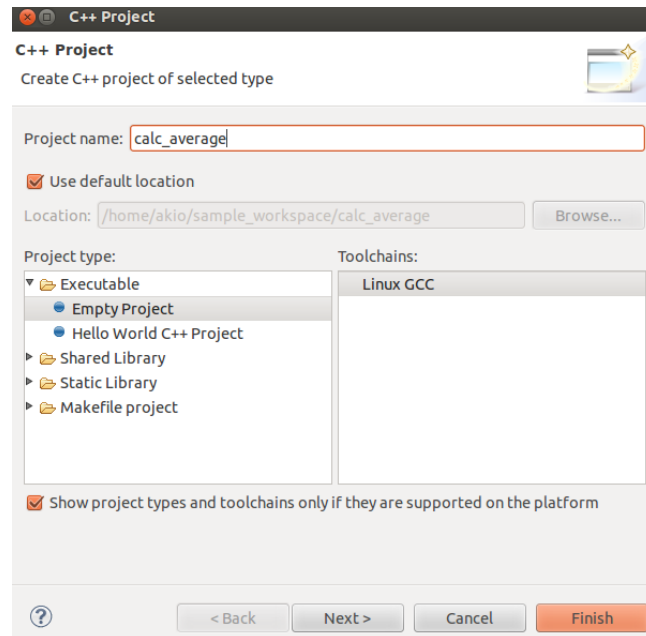
## 簡単なプログラム作成 – プロジェクト作成

Eclipseインストール

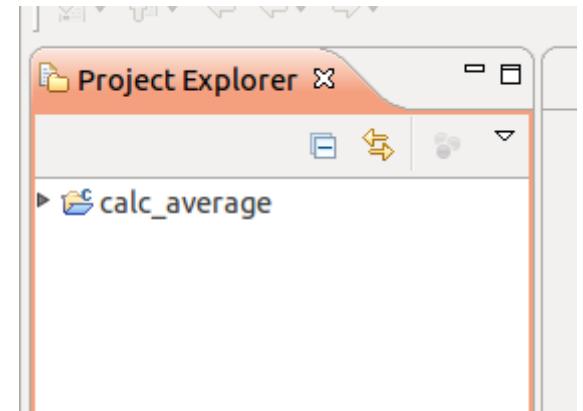
使い方

OpenFOAMカスタマイズ

1. メニュー「File」⇒「New」⇒「C++ Project」  
「C++ Project」ウィザードダイアログ起動
2. 「Project name」に名前を入力、「Empty Project」選択して「Finish」
3. 「Project Explorer」に新しいプロジェクトが作成される



**「C++ Project」ウィザードダイアログ**  
ここではProject nameを「calc\_average」に設定



**Project Explorer**  
デフォルトではワークベンチ左側(\*)  
プロジェクト作成された事がわかる

(\*) 見つからない場合、  
メニュー「Window」⇒  
「Show View」⇒「Project Explorer」で表示

# 使い方

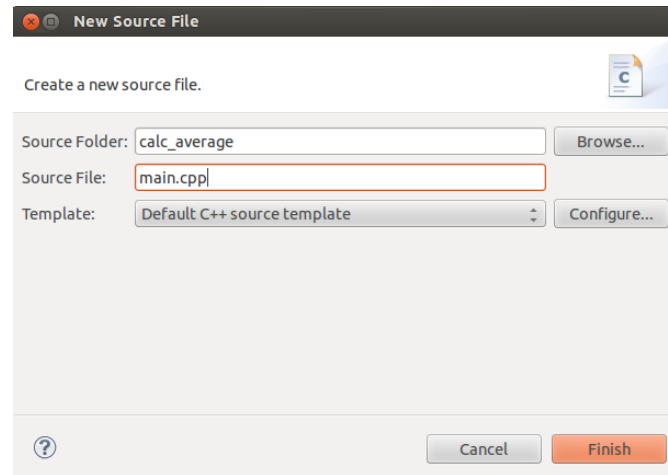
## 簡単なプログラム作成 – ソースコードファイル作成

Eclipseインストール

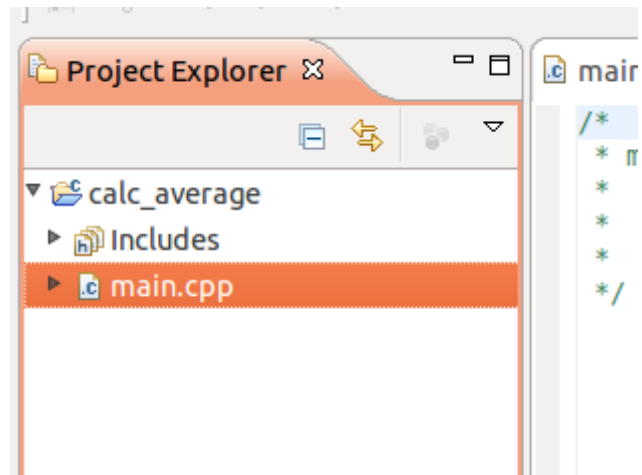
使い方

OpenFOAMカスタマイズ

1. 「Project Explorer」の「calc\_average」を右クリック
2. 「New」⇒「Source File」  
「New Source File」ウィザードダイアログ起動
3. 「Source File」にファイル名を入力して「Finish」
4. 「Project Explorer」の「calc\_average」以下に新しいファイルが作成される



**「New Source File」ウィザードダイアログ**  
ここでは「main.cpp」を入力



**Project Explorer**  
「calc\_average」以下に「main.cpp」が  
作成された事がわかる

# 使い方

## 簡単なプログラム作成 – コーディング

Eclipseインストール

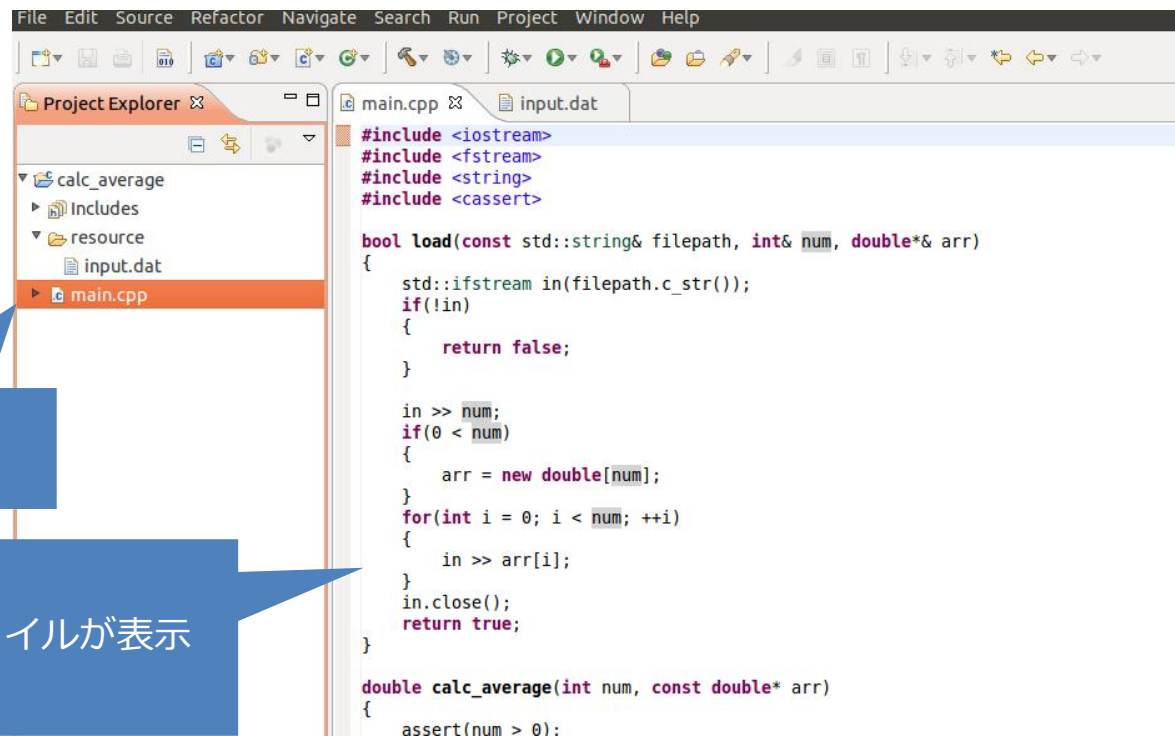
使い方

OpenFOAMカスタマイズ

1. 「Project Explorer」の「main.cpp」をダブルクリック
2. エディタを利用してコーディング

①ダブルクリック

②エディタに  
ソースコードファイルが表示  
コーディング



```
File Edit Source Refactor Navigate Search Run Project Window Help
Project Explorer
  calc_average
    Includes
    resource
      input.dat
      main.cpp
main.cpp
input.dat
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

bool load(const std::string& filepath, int& num, double*& arr)
{
    std::ifstream in(filepath.c_str());
    if(!in)
    {
        return false;
    }

    in >> num;
    if(0 < num)
    {
        arr = new double[num];
    }
    for(int i = 0; i < num; ++i)
    {
        in >> arr[i];
    }
    in.close();
    return true;
}

double calc_average(int num, const double* arr)
{
    assert(num > 0);
```

# 使い方

## 簡単なプログラム作成 – コーディング

Eclipseインストール

使い方

OpenFOAMカスタマイズ

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

bool load(const std::string& filepath, int& num,
double*& arr){
    std::ifstream in(filepath.c_str());
    if(!in){
        return false;
    }
    in >> num;
    if(0 < num){
        arr = new double[num];
    }
    for(int i = 0; i < num; ++i){
        in >> arr[i];
    }
    in.close();
    return true;
}

double calc_average(int num, const double* arr){
    assert(num > 0);
    double sum = 0;
    for(int i = 0; i < num; ++i){
        sum += arr[i];
    }
    return sum / num;
}
```

```
int main(int argc, char* argv[]){
    if(2 > argc){
        std::cout << "error: invalid argument\n";
        return 0;
    }

    int num = 0;
    double* arr = 0;
    if(!load(argv[1], num, arr)){
        std::cout << "error: fail to load, " << argv[1] << "\n";
        return 0;
    }

    if(1 > num){
        std::cout << "error: the number of elements is 0\n";
        return 0;
    }

    std::cout << "num: " << num << "\n";
    std::cout << "average: " << calc_average(num, arr) <<
    "\n";

    delete [] arr;
    return 0;
}
```

# 使い方

## 簡単なプログラム作成 – コーディング

Eclipseインストール

使い方

OpenFOAMカスタマイズ

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
```

```
bool load(const std::string& filepath, int& num,
double*& arr){
    std::ifstream in(filepath.c_str());
    if(!in){
        return false;
    }
    in >> num;
    if(0 < num){
        arr = new double[num];
    }
    for(int i = 0; i < num; ++i){
        in >> arr[i];
    }
    in.close();
    return true;
}
```

```
double calc_average(int num, const double* arr){
    assert(num > 0);
    double sum = 0;
    for(int i = 0; i < num; ++i){
        sum += arr[i];
    }
    return sum / num;
}
```

```
int main(int argc, char* argv[]){
    if(2 > argc){
        std::cout << "error: invalid argument\n";
        return 0;
    }

    int num = 0;
    double* arr = 0;
    if(!load(argv[1], num, arr)){
        std::cout << "error: failed to load " << argv[1] << "\n";
        return 0;
    }
    if(num <= 0){
        std::cout << "error: the number of elements is 0\n";
        return 0;
    }

    std::cout << "num: " << num << "\n";
    std::cout << "average: " << calc_average(num, arr) <<
    "\n";
}
```

数列ファイルを読み込んで、  
数列の要素数(num)と  
数列の要素を格納した  
doubleの配列(arr)を生成

数列の要素数(num)と  
数列の要素を格納した  
doubleの配列(arr)から  
平均値を計算

# 使い方

## 簡単なプログラム作成 - コーディング

Eclipseインストール

使い方

OpenFOAMカスタマイズ

```
#include <iostream>
```

```
#include <fstream>
```

```
#引数の数をチェック & エラー処理
```

```
#include <cassert>
```

```
bool load(const std::string& filepath, int& num,
```

```
double*& arr){
```

```
    std::ifstream in(filepath.c_str());
```

```
    if(!in){
```

```
        return false;
```

```
    }
```

```
    in >> num;
```

```
    if(0 <= num){
```

```
        arr = new double[num];
```

```
        数列要素数が0以下はエラー終了
```

```
        for(int i = 0; i < num; ++i){
```

```
            in >> arr[i];
```

```
        }
```

```
    }
```

```
    in.close();
```

```
    }
```

```
double calc_average(int num, const double* arr){
```

```
    assert(num > 0);
```

```
    double sum = 0;
```

```
    for(int i = 0; i < num; ++i){
```

```
        sum += arr[i];
```

```
    }
```

```
    return sum / num;
```

```
    }
```

```
int main(int argc, char* argv){
```

```
    if(2 > argc){
```

```
        std::cout << "error: invaid argument\n";
```

```
        return 0;
```

```
    }
```

```
    int num = 0;
```

```
    double* arr = 0;
```

```
    if(!load(argv[1], num, arr)){
```

```
        std::cout << "error: fail to load, " << argv[1] << "\n";
```

```
        return 0;
```

```
    }
```

```
    }
```

```
    if(1 > num){
```

```
        std::cout << "error: the number of elements is 0\n";
```

```
        return 0;
```

```
    }
```

```
    }
```

```
    std::cout << "num: " << num << "\n";
```

```
    std::cout << "average: " << calc_average(num, arr) <<
```

```
    "\n";
```

```
    }
```

```
    delete [] arr;
```

```
    return 0;
```

```
    }
```



# 使い方

## Eclipseを使った簡単なソフトウェア開発の流れ



# 使い方

## 簡単なプログラム作成 – デバッグビルド

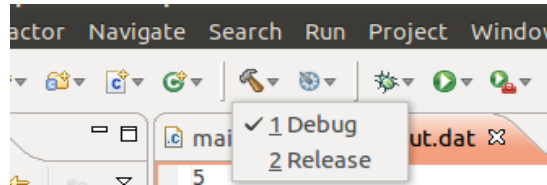
Eclipseインストール

使い方

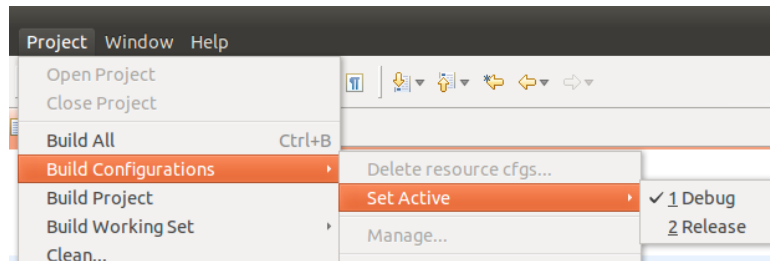
OpenFOAMカスタマイズ

### 1. 「Build Configurations」を「Debug」に設定

方法1：ツールバーのハンマーマーク横の▼を押下



方法2：メニュー「Project」⇒「Build Configurations」⇒「Set Active」



### 2. ビルド

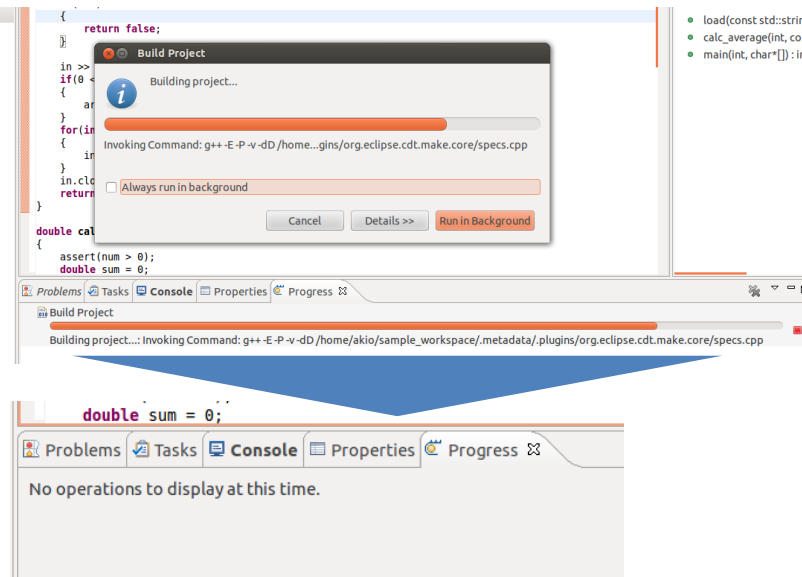
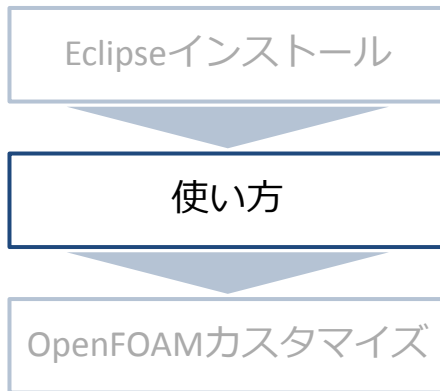
方法1：ツールバーのハンマーマークを押下

方法2：メニュー「Project」⇒「Build All」

# 使い方

## 簡単なプログラム作成 – デバッグビルド

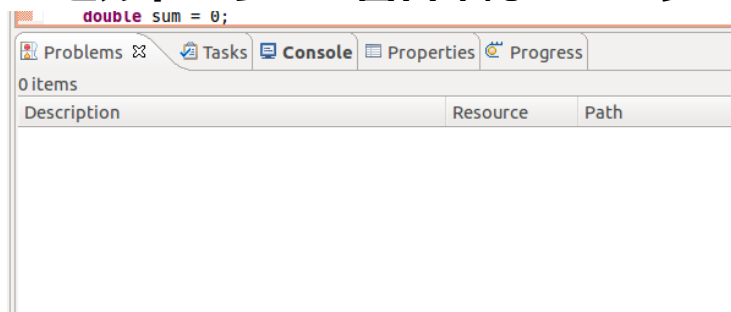
### 3. ビルド完了待ち



「Build Project」ダイアログ  
もしくは「Progress」ウィンドウ  
のプログレスバーが  
100%になるまで待つ

「Progress」ウィンドウの表示が  
「No operations to display at this  
time」となったらビルド完了

### 4. ビルドエラー・警告確認 ⇒ ソースコード修正



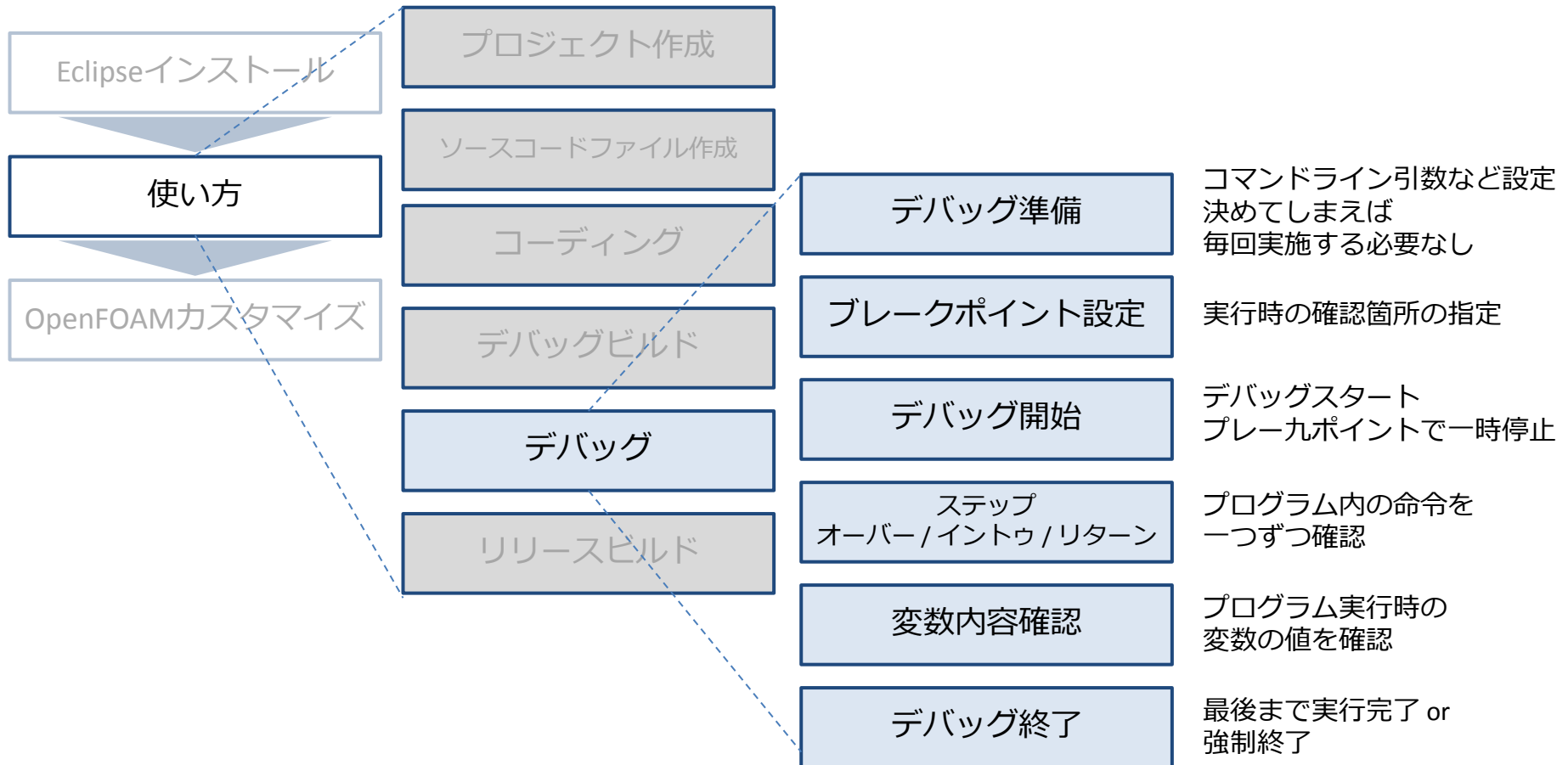
「Problems」ウィンドウに  
エラー項目がなくなったら  
デバッグ準備完了。

エラー項目がある場合、  
ソースコードを修正して  
再ビルド

(\*)  
「Progress」「Problems」ウィンドウは  
メニュー「Window」⇒「Show View」  
で表示

# 使い方

## Eclipseを使った簡単なソフトウェア開発の流れ



# 使い方

## 簡単なプログラム作成 – デバッグ

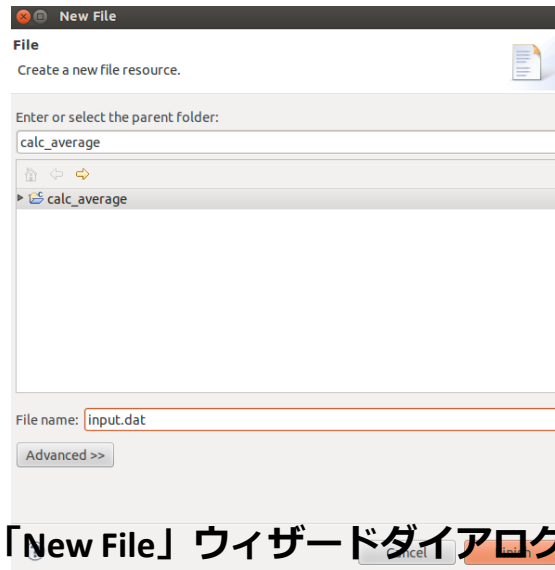
### 1. デバッグ準備 – 入力ファイルの作成

1. 「Project Explorer」の「calc\_average」を右クリック
2. 「New」⇒「File」  
「New File」ウィザードダイアログ起動
3. 「File name」にファイル名を入力して「Finish」
4. 「Project Explorer」の「calc\_average」以下に新しいファイルが作成される

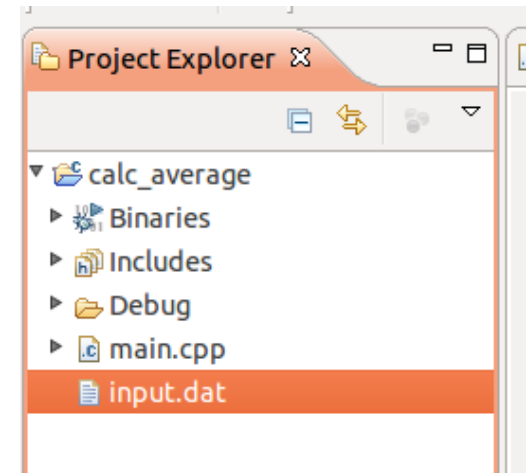
Eclipseインストール

使い方

OpenFOAMカスタマイズ



「New File」ウィザードダイアログ  
ここでは「input.dat」を入力



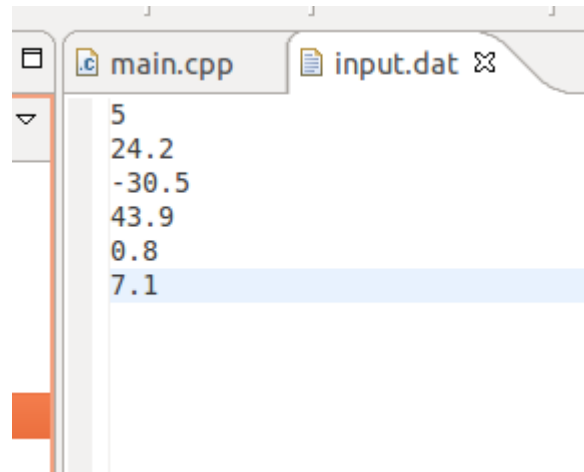
Project Explorer  
「calc\_average」以下に「input.dat」が  
作成された事がわかる

# 使い方

## 簡単なプログラム作成 – デバッグ

### 1. デバッグ準備 – 入力ファイルの編集

5. 「Project Explorer」の「input.dat」をダブルクリック  
エディタにinput.datのファイル内容が表示
6. エディタを利用して編集



The screenshot shows the Eclipse IDE's editor window with two tabs: 'main.cpp' and 'input.dat'. The 'input.dat' tab is active, displaying the following text:

```
5
24.2
-30.5
43.9
0.8
7.1
```

### エディタで「input.dat」の編集

ここでは15ページの例に従って作成

Eclipseインストール

使い方

OpenFOAMカスタマイズ

# 使い方

## 簡単なプログラム作成 – デバッグ

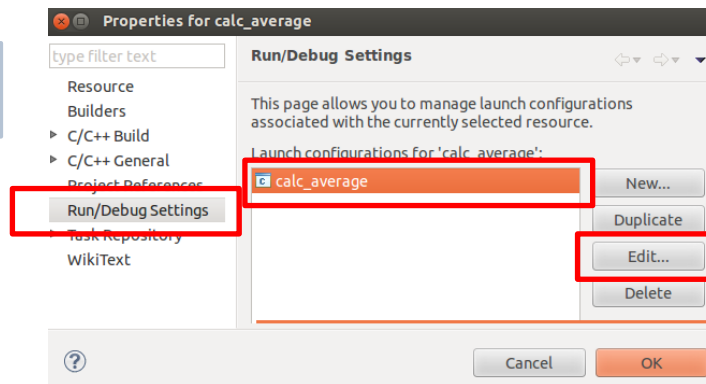
Eclipseインストール

使い方

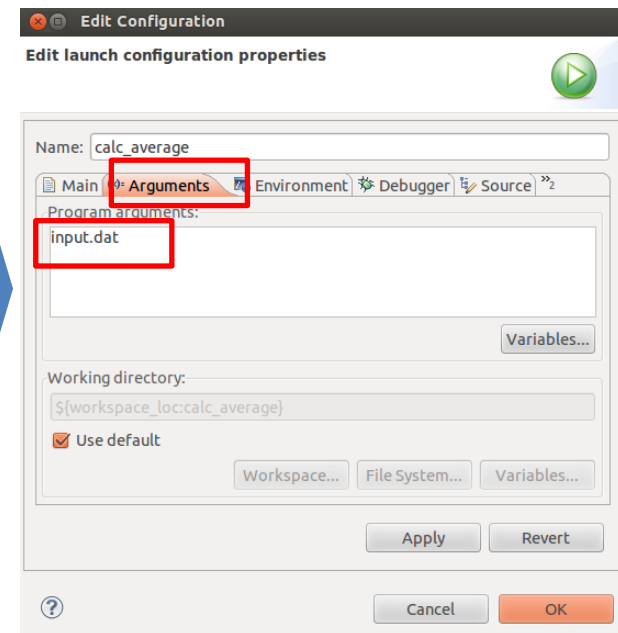
OpenFOAMカスタマイズ

### 1. デバッグ準備 – デバッグ時のコマンドライン引数の設定

7. 「Project Explorer」の「calc\_average」を右クリック
8. 「Properties」でcalc\_averageの設定ダイアログ起動
9. 「Run/Debug Settings」でcalc\_averageを選択し「edit」
10. 「Arguments」タブの「Program arguments」に「inputs.dat」



calc\_averageの設定ダイアログ  
「input.dat」を入力

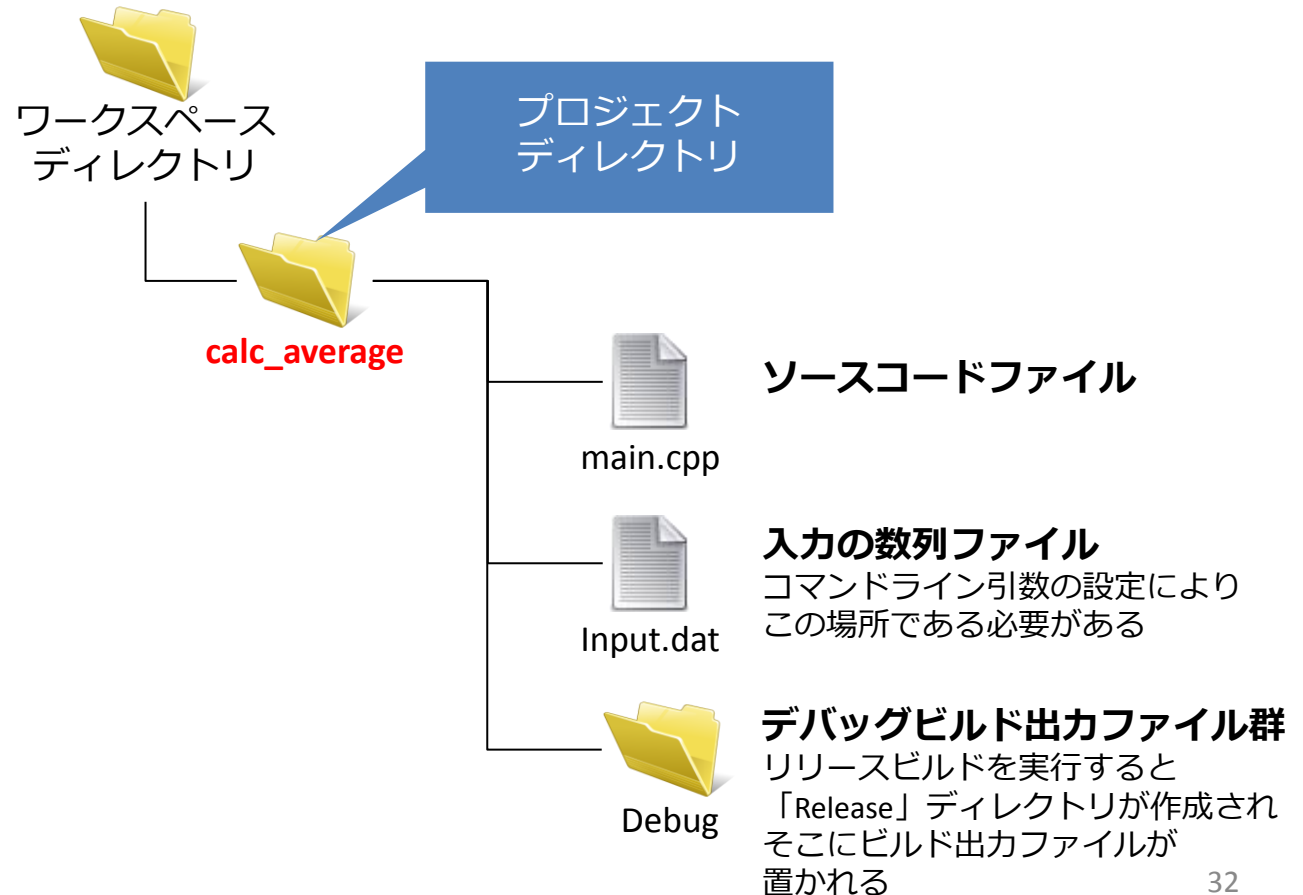
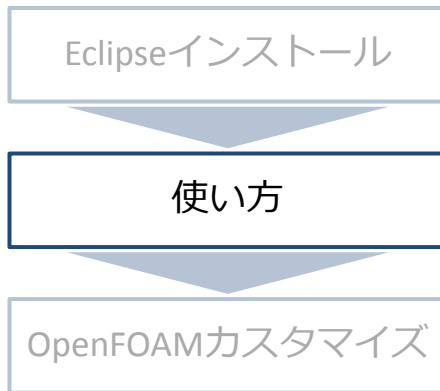


コマンドライン引数を設定

# 使い方

## 簡単なプログラム作成 – デバッグ

### ワークスペース内のディレクトリ・ファイル構成の確認





# 使い方

## 簡単なプログラム作成 – デバッグ

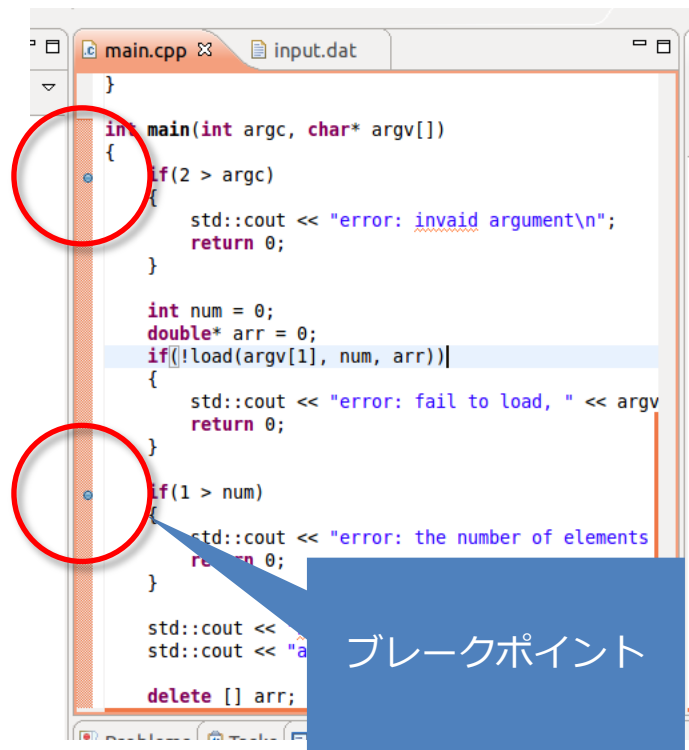
### 2. ブレークポイント設定

デバッグ開始すると自動的にブレークポイントで停止

Eclipseインストール

使い方

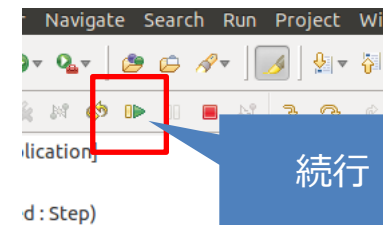
OpenFOAMカスタマイズ



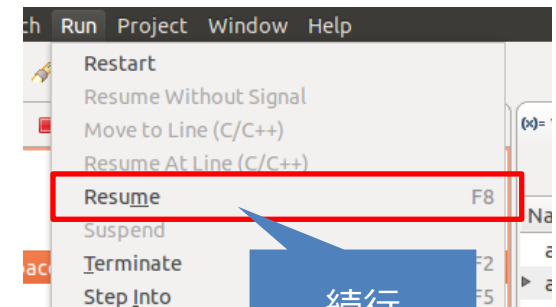
```
main.cpp | input.dat
}
int main(int argc, char* argv[])
{
    if(2 > argc)
    {
        std::cout << "error: invalid argument\n";
        return 0;
    }
    int num = 0;
    double* arr = 0;
    if(!load(argv[1], num, arr))
    {
        std::cout << "error: fail to load, " << argv
        return 0;
    }
    if(1 > num)
    {
        std::cout << "error: the number of elements
        return 0;
    }
    std::cout <<
    std::cout << "a
    delete [] arr;
```

ブレークポイント

エディタ左側をダブルクリックで  
ブレークポイント作成 / 削除



続行



続行

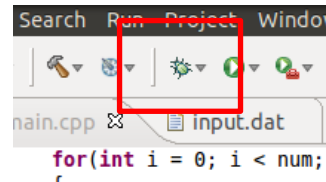
一時停止状態を解除するには  
ツールバー「続行」ボタン or  
メニュー「Run」⇒「Resume」  
(Debugパースペクティブ)

# 使い方

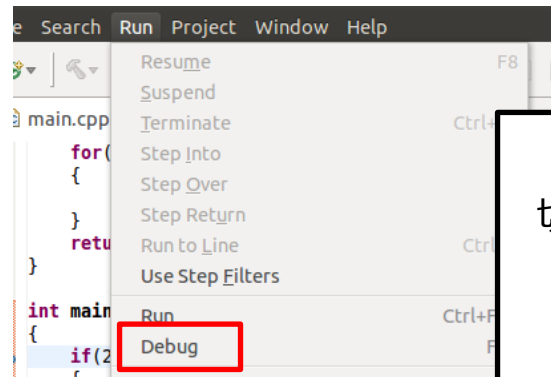
## 簡単なプログラム作成 – デバッグ

### 3. デバッグ開始

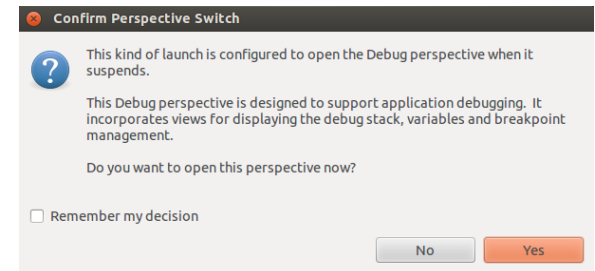
方法1：ツールバーの虫マークを押下



方法2：メニュー「Run」⇒「Debug」



Debugパースペクティブに切り替わる警告が表示。とりあえずYes



Eclipseインストール

使い方

OpenFOAMカスタマイズ

# 使い方

## 簡単なプログラム作成 – デバッグ

### 4. ステップ オーバー / イントウ / リターン

プログラム内の命令を一つずつ確認

ブレークポイントにより一時停止中に利用するデバッグ機能

- ステップオーバー：  
一時停止中の命令を一つ進める  
命令が関数の場合、関数内に入らず、再度一時停止
- ステップイントウ：  
一時停止中の命令を一つ進める  
命令が関数の場合、関数内に入り、再度一時停止
- ステップリターン：  
一時停止している関数を実行完了したところで、  
再度一時停止

Eclipseインストール

使い方

OpenFOAMカスタマイズ

# 使い方

## 簡単なプログラム作成 – デバッグ

### 4. ステップ オーバー / イントウ / リターン

#### ステップオーバーの確認 :

main関数内の関数load命令にブレークポイントを設定しデバッグ  
(最初からデバッグ)

Eclipseインストール

使い方

OpenFOAMカスタマイズ

ブレークポイント

```
    }  
    return sum / num;  
}  
  
int main(int argc, char* argv[])  
{  
    if(2 > argc)  
    {  
        std::cout << "error: invalid argument\n";  
        return 0;  
    }  
  
    int num = 0;  
    double* arr = 0;  
    if(!load(argv[1], num, arr))  
    {  
        std::cout << "error: fail to load, " << a  
        return 0;  
    }  
  
    if(1 > num)
```

- ①関数load実行前で一時停止
- ②ステップオーバー実行

```
    }  
    return sum / num;  
}  
  
int main(int argc, char* argv[])  
{  
    if(2 > argc)  
    {  
        std::cout << "error: invalid argument\n";  
        return 0;  
    }  
  
    int num = 0;  
    double* arr = 0;  
    if(!load(argv[1], num, arr))  
    {  
        std::cout << "error: fail to load, " << a  
        return 0;  
    }  
  
    if(1 > num)  
    {  
        std::cout << "error: the number of elemen  
        return 0;  
    }  
}
```

- ③main関数内の次の命令へ制御移動  
(関数loadがtrueを返却した場合)  
(main関数内で一時停止)

# 使い方

## 簡単なプログラム作成 – デバッグ

### 4. ステップ オーバー/イントウ/リターン

#### ステップイントウの確認 :

main関数内の関数load命令にブレークポイントを設定しデバッグ  
(最初からデバッグ)

Eclipseインストール

使い方

OpenFOAMカスタマイズ

ブレークポイント

```
    }  
    return sum / num;  
}  
  
int main(int argc, char* argv[])  
{  
    if(2 > argc)  
    {  
        std::cout << "error: invalid argument\n";  
        return 0;  
    }  
  
    int num = 0;  
    double* arr = 0;  
    if(!load(argv[1], num, arr))  
    {  
        std::cout << "error: fail to load, " << a  
        return 0;  
    }  
  
    if(1 > num)
```

```
#include <iostream>  
#include <fstream>  
#include <string>  
#include <cassert>  
  
bool load(const std::string& filepath, int& num,  
{  
    std::ifstream in(filepath.c_str());  
    if(!in)  
    {  
        return false;  
    }  
  
    in >> num;  
    if(0 < num)  
    {  
        arr = new double[num];  
    }  
    for(int i = 0; i < num; ++i)  
    {  
        in >> arr[i];  
    }  
    in.close();  
    return true;  
}
```

- ①関数load実行前で一時停止
- ②ステップイントウ実行

- ③関数load内へ制御移動  
(関数load内で一時停止)

# 使い方

## 簡単なプログラム作成 – デバッグ

### 4. ステップ オーバー/イントウ/リターン

**ステップリターンの確認：**  
前ページの③の状態からデバッグ  
(関数load内で一時停止状態)

Eclipseインストール

使い方

OpenFOAMカスタマイズ

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

bool load(const std::string& filepath, int& num,
{
    std::ifstream in(filepath.c_str());
    if(!in)
    {
        return false;
    }

    in >> num;
    if(0 < num)
    {
        arr = new double[num];
    }
    for(int i = 0; i < num; ++i)
    {
        in >> arr[i];
    }
    in.close();
    return true;
}
```

- ①関数load内で一時停止
- ②ステップリターン実行

```
}
return sum / num;
}

int main(int argc, char* argv[])
{
    if(2 > argc)
    {
        std::cout << "error: fail to load, " << a
        return 0;
    }

    int num = 0;
    double* arr = 0;
    if(!load(argv[1], num, arr))
    {
        std::cout << "error: fail to load, " << a
        return 0;
    }

    if(1 > num)
    {
        return sum / num;
    }
}
```

関数loadが実行され  
Main関数に戻ってくる

- ③呼び出し元であるmain関数  
へ制御移動  
(main関数で一時停止)

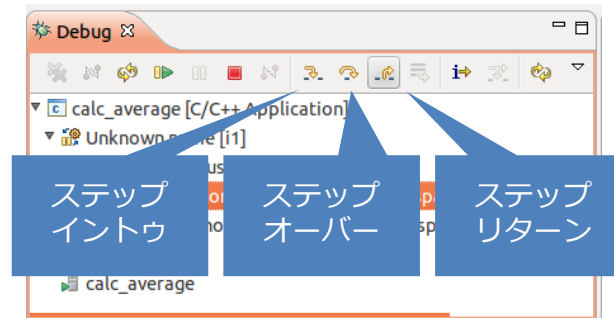
# 使い方

## 簡単なプログラム作成 – デバッグ

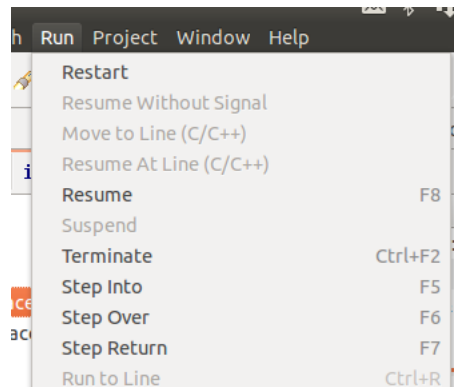
### 4. ステップ オーバー/イントウ/リターン

#### 実行方法

方法1 : 「Debug」 ウィンドウ (スタックトレース) のボタン



方法2 : メニュー 「Run」 ⇒ 「Step Over」 「Step Into」 「Step Return」



Eclipseインストール

使い方

OpenFOAMカスタマイズ

# 使い方

## 簡単なプログラム作成 – デバッグ

### 5. 変数内容確認

「Variables」ウィンドウ：

一時停止中時に利用されている変数を自動判別、値表示  
ローカル変数などの確認に有効

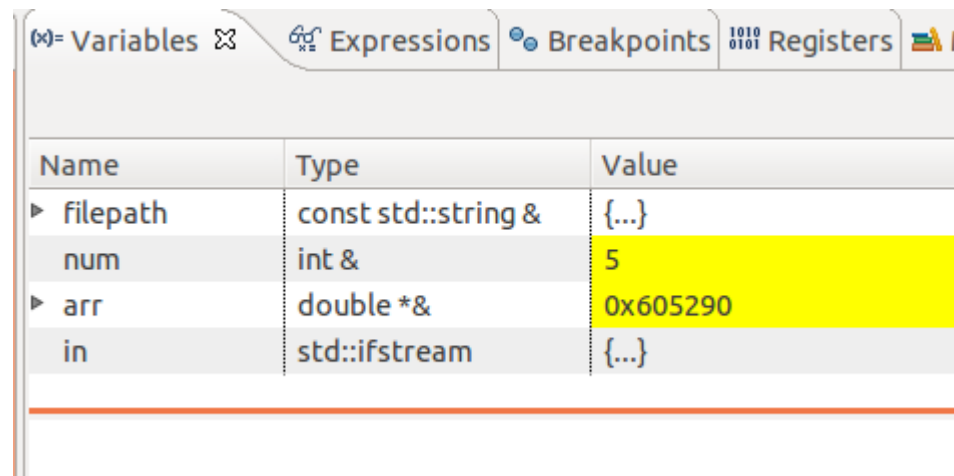
「Expressions」ウィンドウ：

自分で値を確認したい変数を指定できる  
グローバル変数の値の確認、  
メンバ関数実行時の同じクラス内  
のメンバ変数の値の確認に有効

Eclipseインストール

使い方

OpenFOAMカスタマイズ



The screenshot shows the Eclipse IDE's Variables window. The window title is "Variables" and it contains a table with the following columns: Name, Type, and Value. The table lists several variables, with the "num" variable highlighted in yellow.

Name	Type	Value
▶ filepath	const std::string &	{...}
num	int &	5
▶ arr	double *&	0x605290
in	std::ifstream	{...}



# 使い方

## 簡単なプログラム作成 – デバッグ

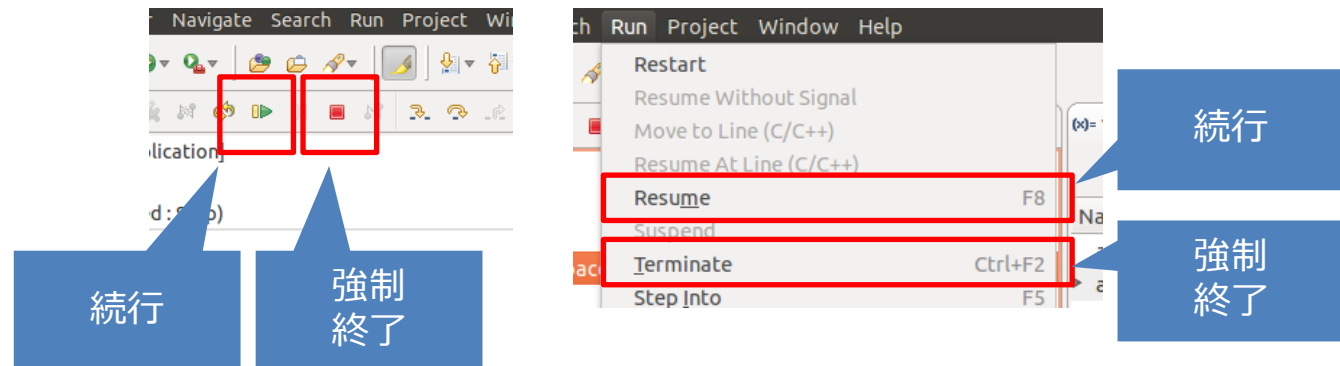
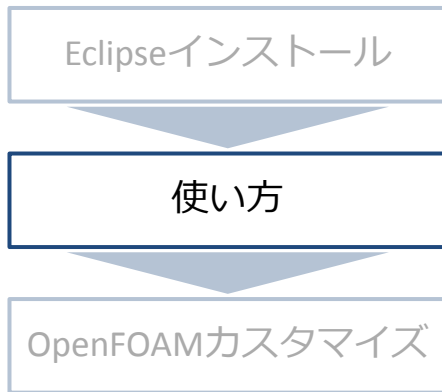
### 6. デバッグ終了方法

方法1：最後まで実行

ツールバーの続行ボタン押下  
メニュー「Run」⇒「Resume」

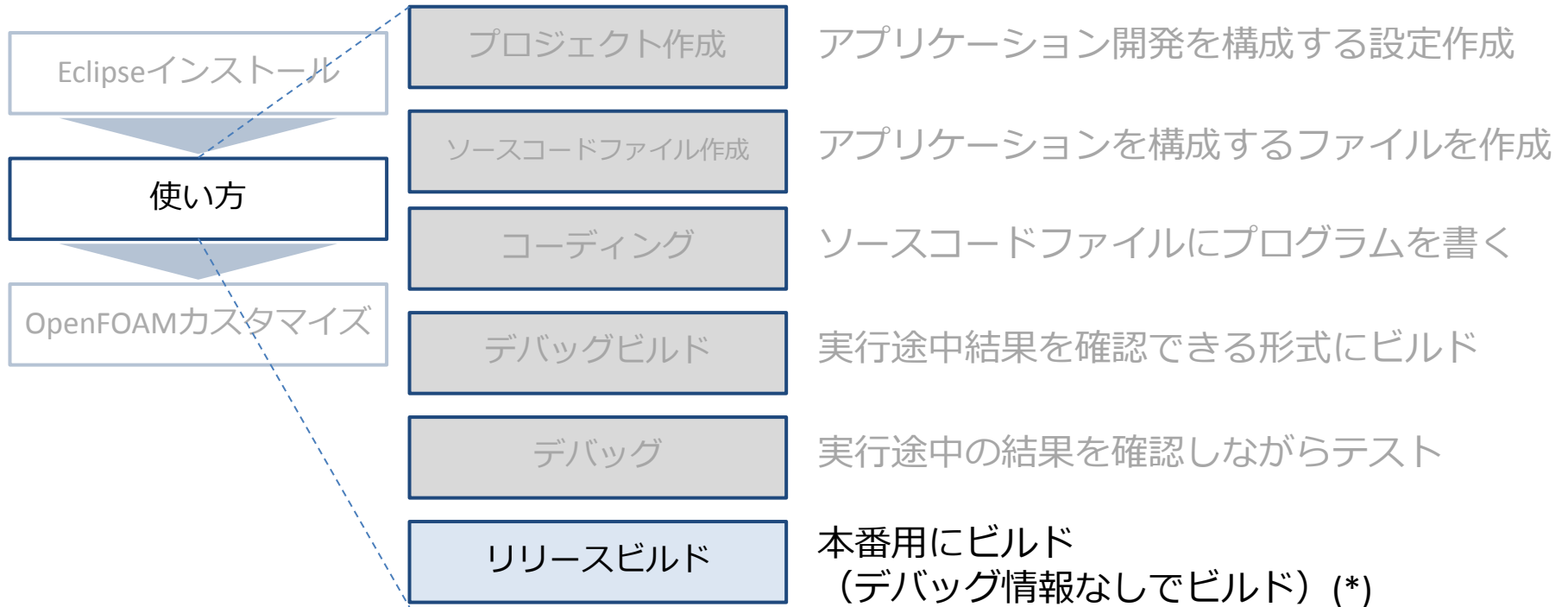
方法2：途中で強制終了

ツールバーの終了ボタン押下  
メニュー「Run」⇒「Terminate」



# 使い方

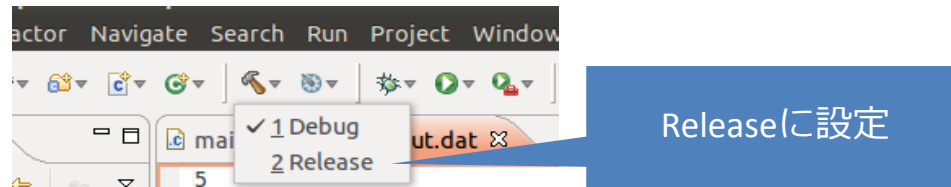
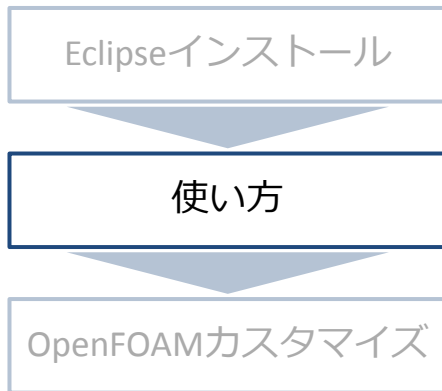
## Eclipseを使った簡単なソフトウェア開発の流れ



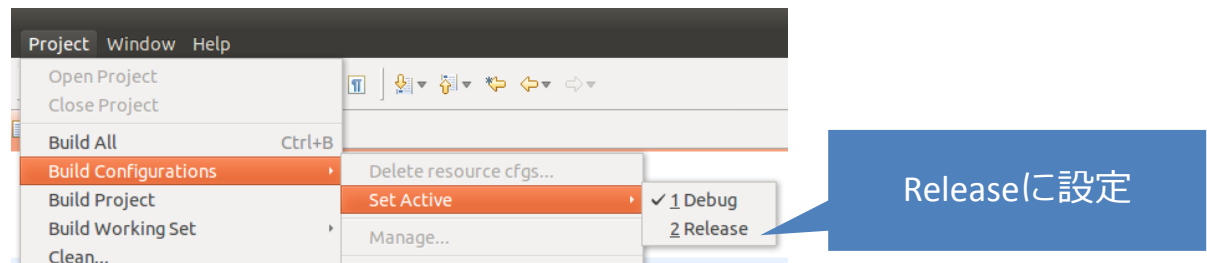
# 使い方

## 簡単なプログラム作成 – リリースビルド

1. 「Build Configurations」を「Release」に設定  
方法1：ツールバーのハンマーマーク横の▼を押下



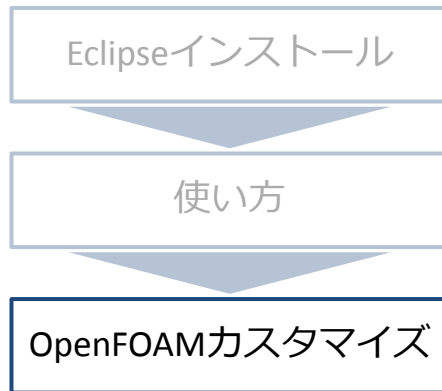
- 方法2：メニュー「Project」⇒「Build Configurations」⇒「Set Active」



2. 以降、デバッグビルドと同様

# OpenFOAMカスタマイズ

前回の例をEclipseで実施



1. 準備
2. Eclipse設定・ビルド
3. デバッグ準備
4. 実行設定
5. 実行

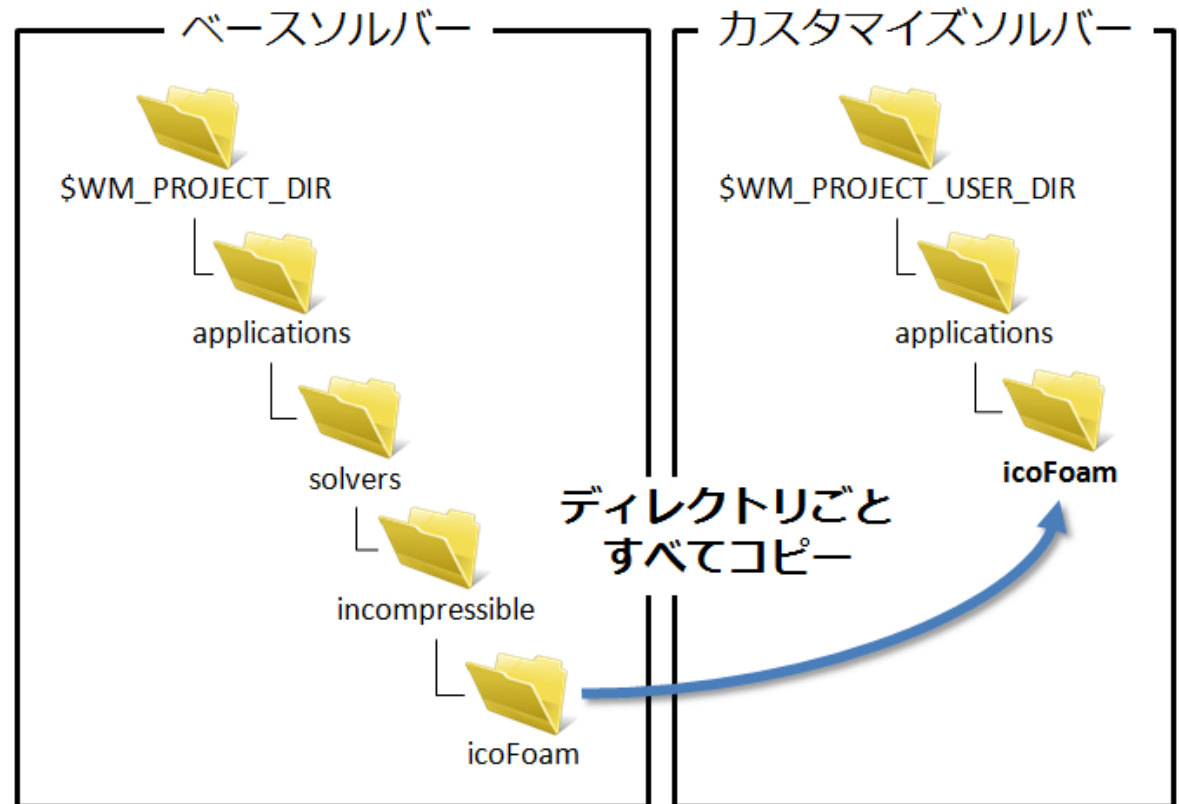
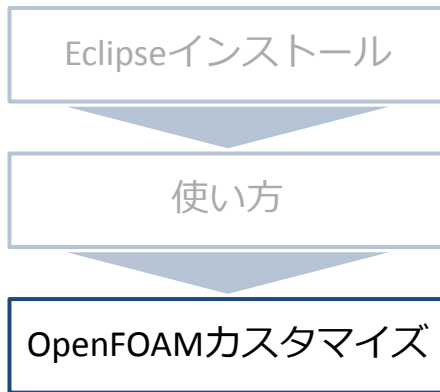
(\*)  
カスタマイズ用ベースソルバーのソースコードの準備、  
ケースファイルの準備は通常と同じやり方（前回参照）

# OpenFOAMカスタマイズ

## 準備

### 1. カスタマイズ用ベースソルバー ソースコード準備

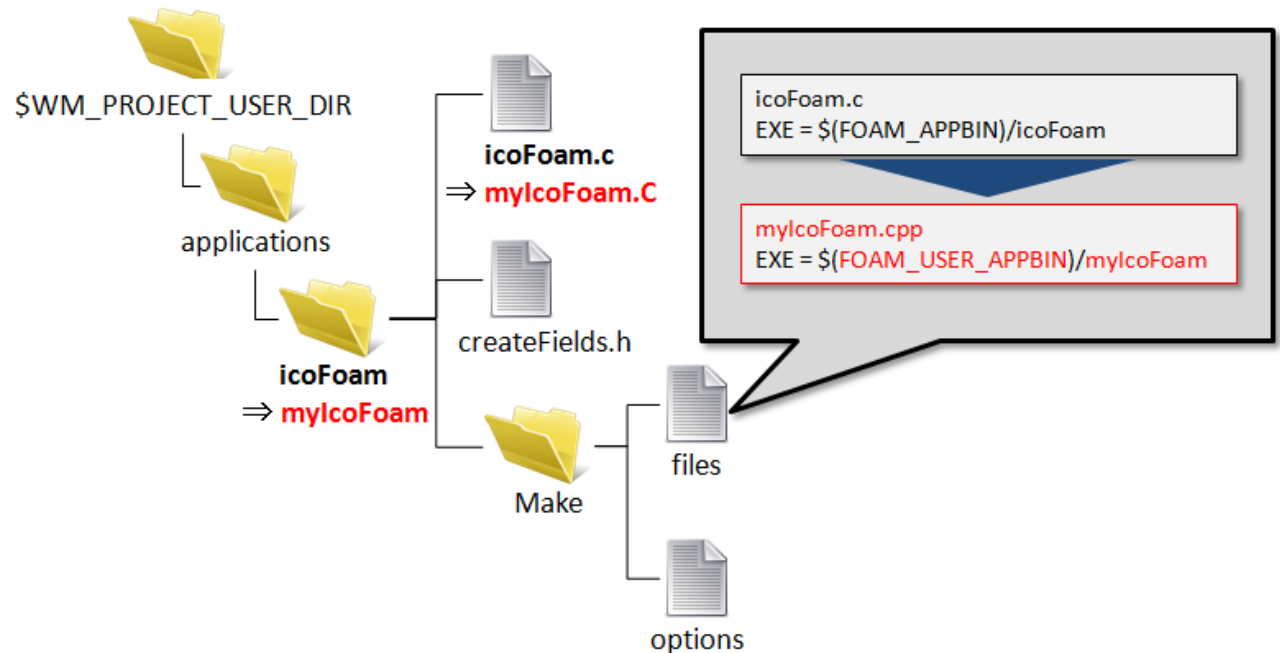
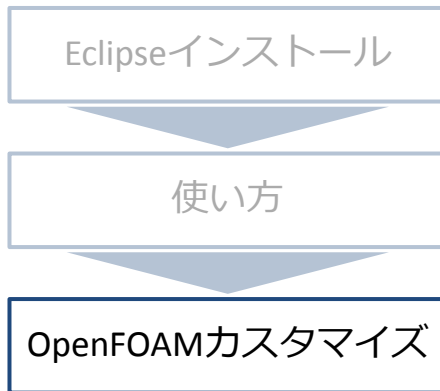
#### 1. ベースソルバーのコピー



# OpenFOAMカスタマイズ

## 準備

1. カスタマイズ用ベースソルバー ソースコード準備
2. ディレクトリ・ファイル名変更



# OpenFOAMカスタマイズ

## 準備

### 2. デバッグビルド / リリースビルド設定 \$WM\_PROJECT\_DIR/etc/bashrcの変更が必要

Eclipseインストール

使い方

OpenFOAMカスタマイズ

```
bashrc (/opt/openfoam211/etc) - gedit
File Edit View Search Tools Documents Help
bashrc *
#- Architecture:
#   WM_ARCH_OPTION = 32 | 64
export WM_ARCH_OPTION=64

#- Precision:
#   WM_PRECISION_OPTION = DP | SP
export WM_PRECISION_OPTION=DP

#- Optimised, debug, profiling:
#   WM_COMPILE_OPTION = Opt | Debug | Prof
# export WM_COMPILE_OPTION=Opt
export WM_COMPILE_OPTION=Debug

#- MPI implementation:
#   WM_MPLIB = SYSTEMOPENMPI | OPENMPI | MPICH | M
#           | GAMMA | MPI | QSMPI
export WM_MPLIB=SYSTEMOPENMPI

#- Operating System:
#   WM_OSTYPE = POSIX | ???
export WM_OSTYPE=POSTY
Plain Text Tab Width: 8
```

79行目あたり

WM\_COMPILE\_OPTIONの値を  
Optにするとリリースビルド  
Debugにするとデバッグビルド  
がwmake時に実行される

### デバッグ / リリースビルドの切替時の注意 :

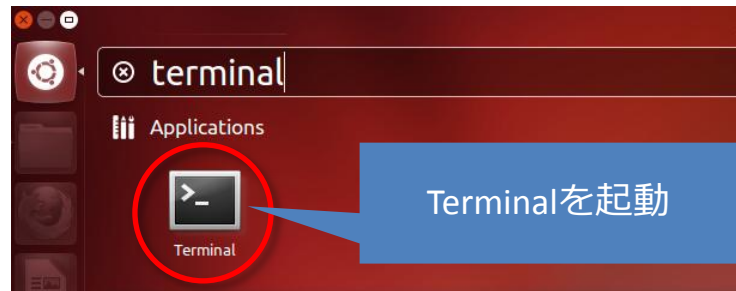
- bashrcの修正、保存後にEclipseを再起動すること (\* 起動方法については後述)
- wmake前にwcleanを実行する事 (\* 実行方法については後述)

# OpenFOAMカスタマイズ

## 準備

### 2. Eclipse起動 – Terminalから起動

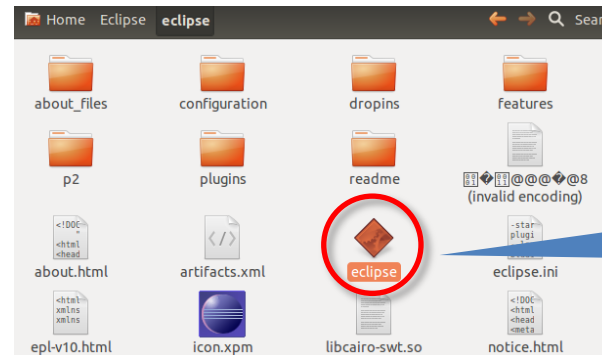
1. 「Dash Home」で“terminal”で検索し、「Terminal」起動(\*)



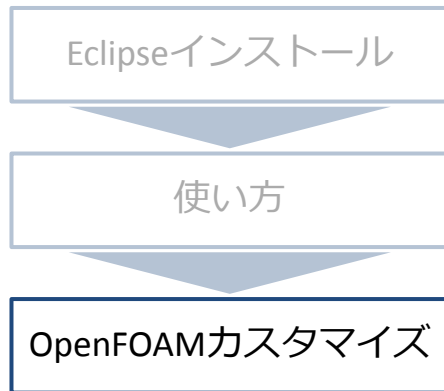
(\*) デバッグ/リリース  
ビルド切替時は  
「Terminal」も再起動の事

### 2. Eclipseを起動

Terminal上で、“[インストールパス]/eclipse”と入力



これをTerminal上で  
起動



### 背景：

OpenFOAMの利用に必要な  
環境変数の設定がシェル起動時に  
設定されるため(\*\*)

(\*\*) 設定についてはリンク参考：<http://www.openfoam.org/download/ubuntu.php>



# OpenFOAMカスタマイズ

## Eclipse設定・ビルド

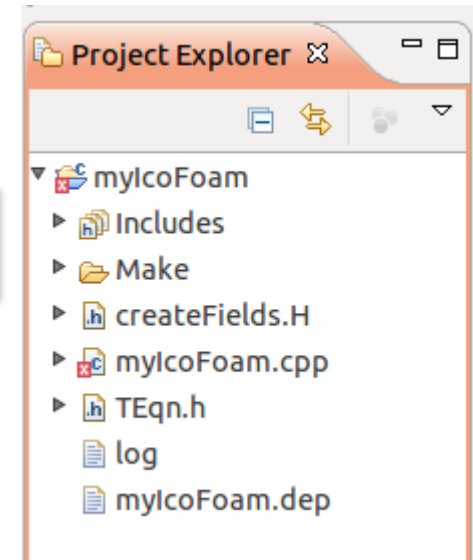
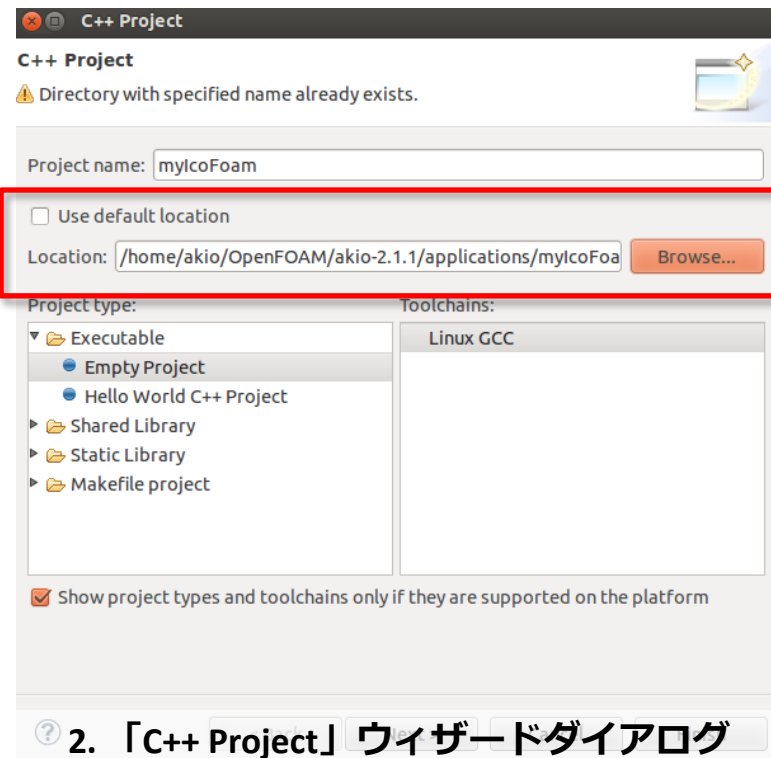
### 1. ソースコード読み込み

1. メニュー「File」⇒「New」⇒「C++ Project」

Eclipseインストール

使い方

OpenFOAMカスタマイズ



**Project Explorer**  
準備したソースコードが  
プロジェクトとしてロード

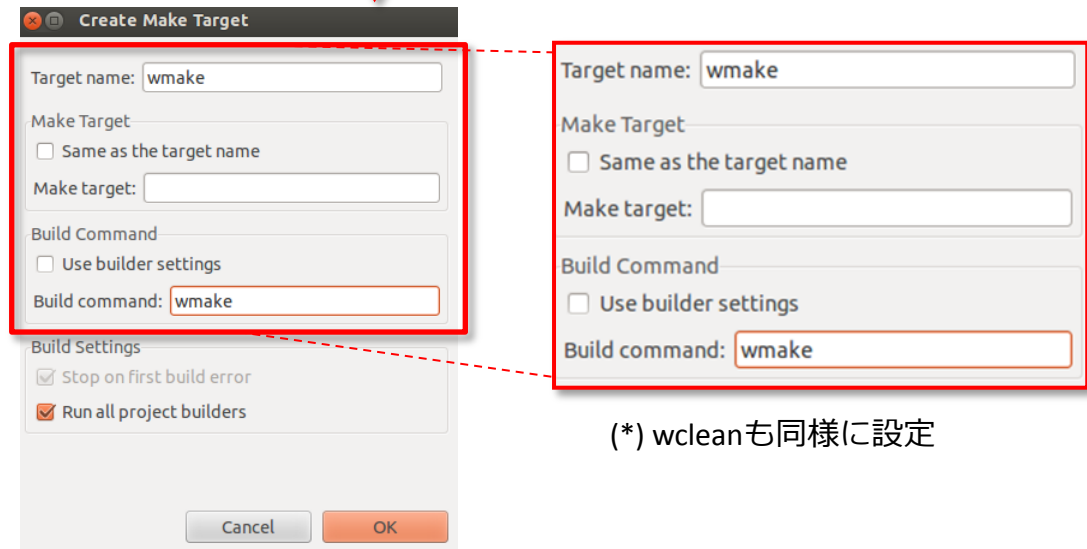
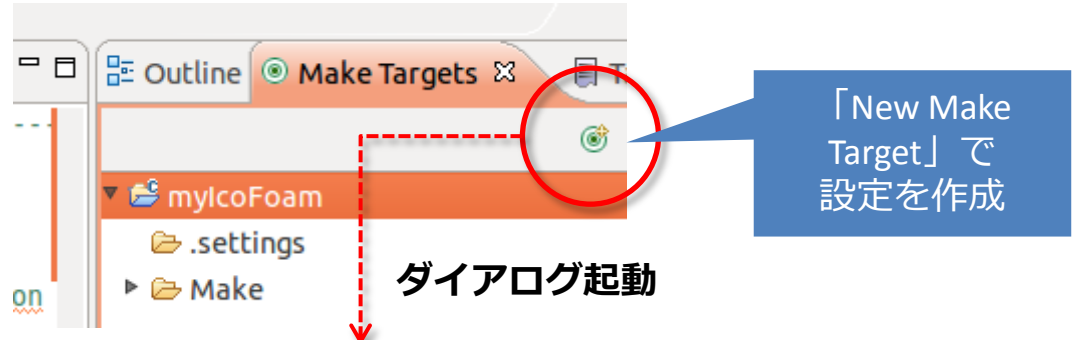
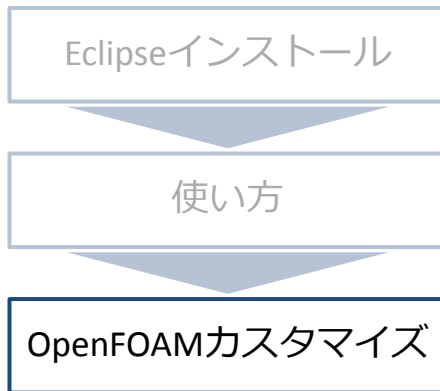
ここではProject nameを「mylcoFoam」に、  
Locationを準備したソルバーソースコードのディレクトリを指定

# OpenFOAMカスタマイズ

## Eclipse設定・ビルド

### 2. wmake / wclean設定

1. 画面右の「Make Targets」を操作してwmakeをwcleanの設定を作成



(\* ) wcleanも同様に設定

# OpenFOAMカスタマイズ

## Eclipse設定・ビルド

### 2. wmake / wclean設定

#### 2. Makefileが置いてある位置を指定

1. 「Project Explorer」の「mylcoFoam」を右クリック
2. 「Properties」をクリック
3. 「mylcoFoam」のプロジェクト設定ダイアログが起動

Eclipseインストール

使い方

OpenFOAMカスタマイズ

C/C++ Build  
Builder type: External builder  
Use default build command  
Build command: make  
Build location  
Build directory: /home/akio/OpenFOAM/akio-2.1.1/applications/mylcoFoam

C/C++ Build  
Build location  
Build directory: /home/akio/OpenFOAM/akio-2.1.1/applications/mylcoFoam

C/C++ Build を選択

準備したソルバーソースコードのディレクトリを指定

# OpenFOAMカスタマイズ

## Eclipse設定・ビルド

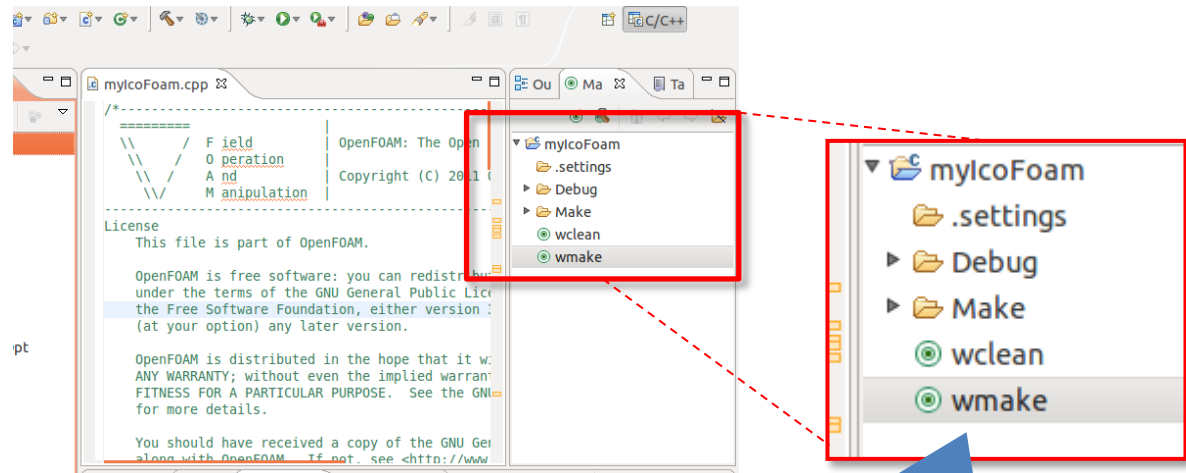
### 3. ビルド

「Make Targets」ウィンドウのwmake, wcleanをダブルクリック

Eclipseインストール

使い方

OpenFOAMカスタマイズ



(\*)  
デバッグ/リリースビルドの設定は  
\$WM\_PROJECT\_DIR/etc/bashrcのみに依存

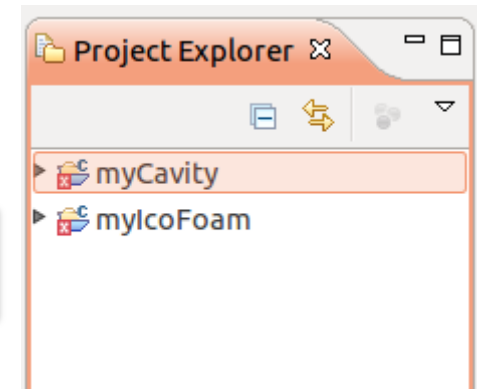
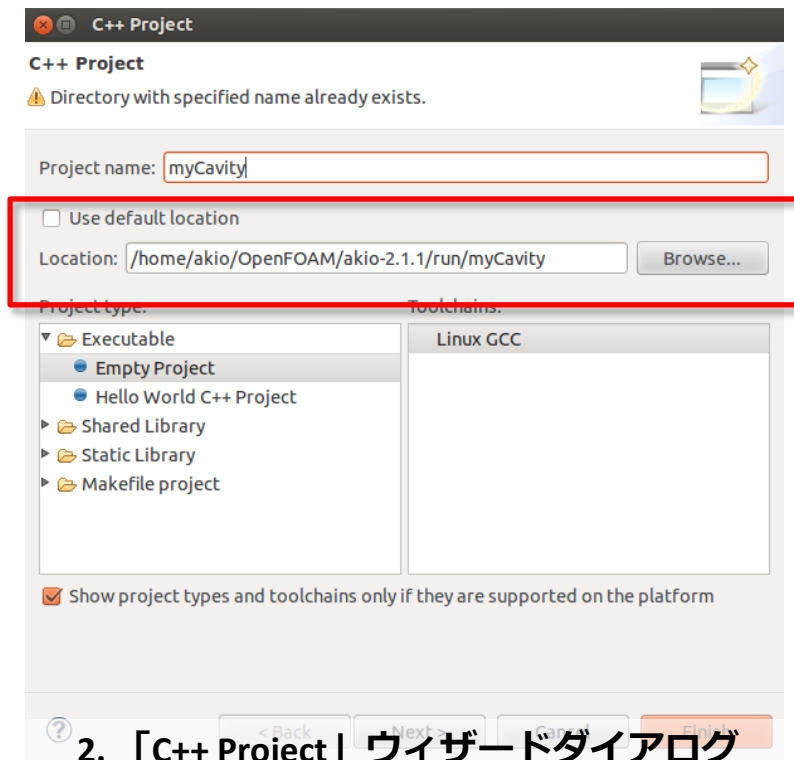
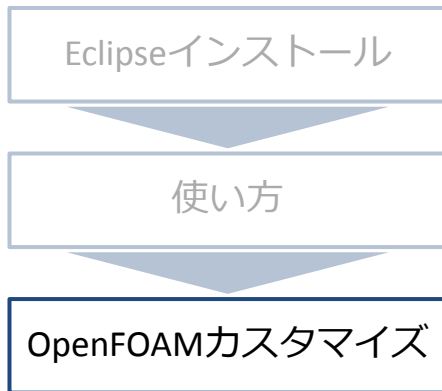
(\*\*)  
デバッグビルド (+ 実行) には、デバッグビルドされたライブラリが必要

# OpenFOAMカスタマイズ

## デバッグ準備

### 1. ケースファイルの設定

1. メニュー「File」⇒「New」⇒「C++ Project」



**Project Explorer**  
ケースファイル群が  
プロジェクトとしてロード

(\*)  
blockMeshは予め  
実行しておく必要あり

2. 「C++ Project」ウィザードダイアログ  
ここではProject nameを「myCavity」に、  
Locationを準備した編集したケースファイルのディレクトリを指定

# OpenFOAMカスタマイズ

## デバッグ準備

### 1. ケースファイルの設定

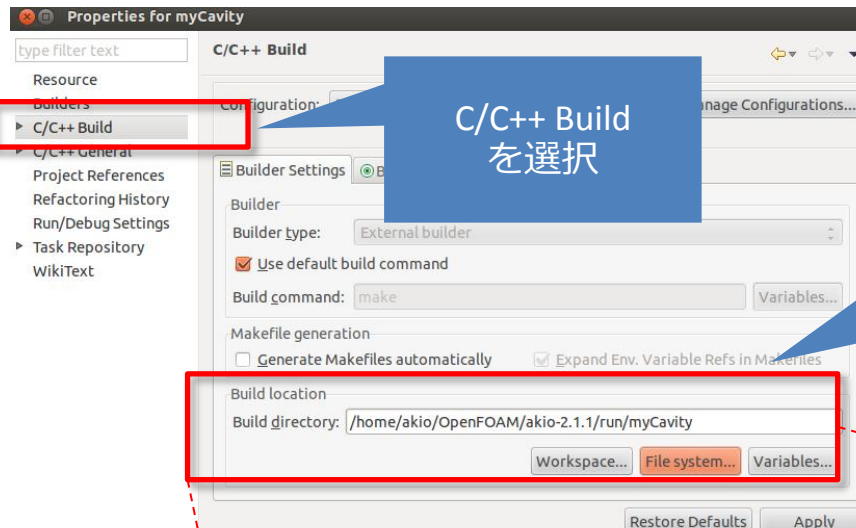
#### 2. Makefileがおいてある位置を指定

1. 「Project Explorer」の「myCavity」を右クリック
2. 「Properties」をクリック
3. 「myCavity」のプロジェクト設定ダイアログが起動

Eclipseインストール

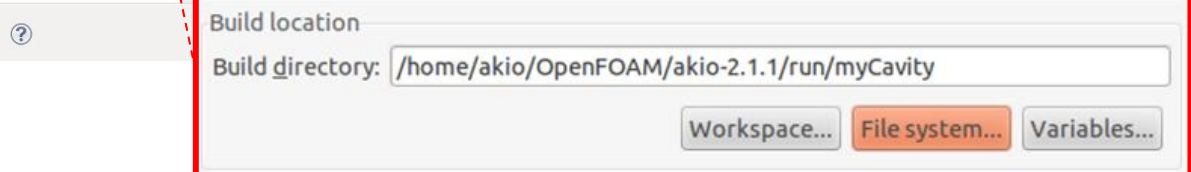
使い方

OpenFOAMカスタマイズ



C/C++ Build  
を選択

準備したケースファイルのディレクトリを指定

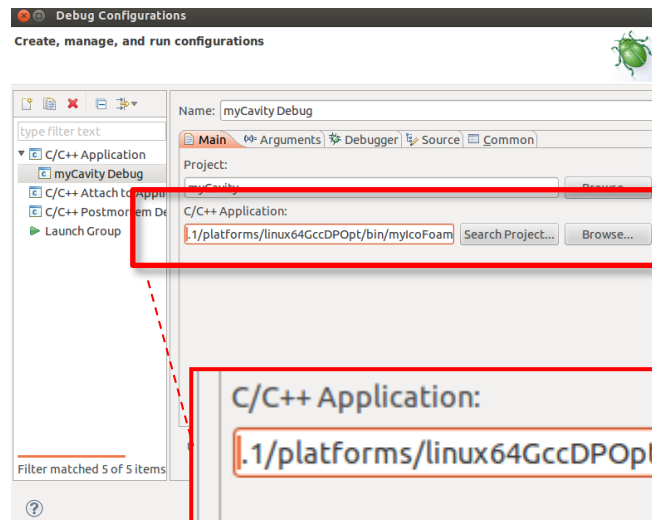
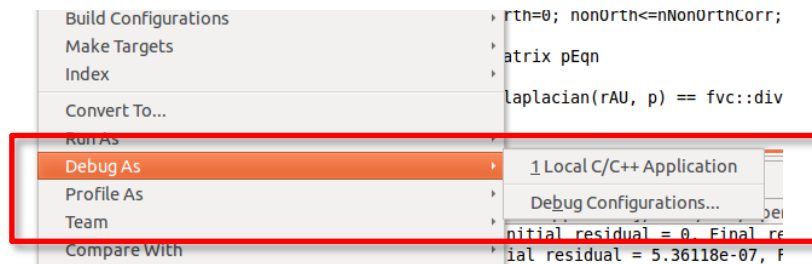
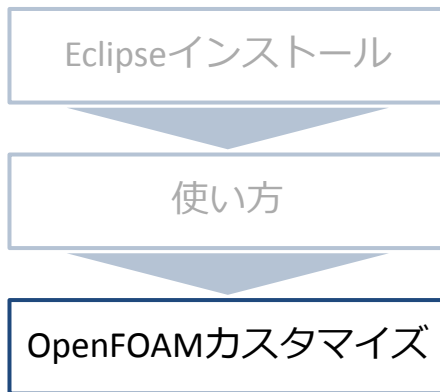


# OpenFOAMカスタマイズ

## 実行設定 - ケースファイルプロジェクトの設定変更

### 1. 実行の設定

1. 「Project Explorer」の「**myCavity**」を右クリック
2. 「Debug As」⇒「Debug Configuration」をクリック



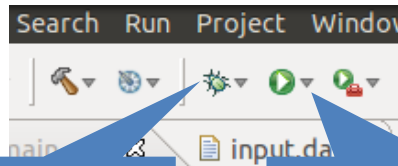
(\*)  
デバッグビルドされた実行形式は  
"\$SWM\_PROJECT\_USER\_DIR/  
platforms/linux64GccDPDebug"  
リリースビルドされた実行形式は  
"\$SWM\_PROJECT\_USER\_DIR/  
platforms/linux64GccDPOpt"  
デバッグ/リリース実行の度に  
設定変更が必要

# OpenFOAMカスタマイズ

## 実行

### 実行：

方法1：ツールバーのボタン



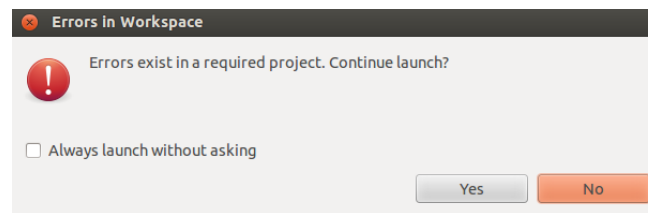
デバッグ

デバッグなし実行

方法2：メニュー「Run」⇒「Debug」or「Run」

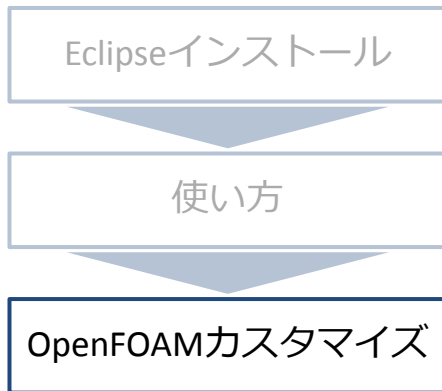
### 実行時エラーダイアログ：

ケースファイルプロジェクトにソースコードがないためビルドエラーが発生  
無視して実行（「Yes」を押下）



### 実行時結果：

Eclipseの「Console」ダイアログに表示





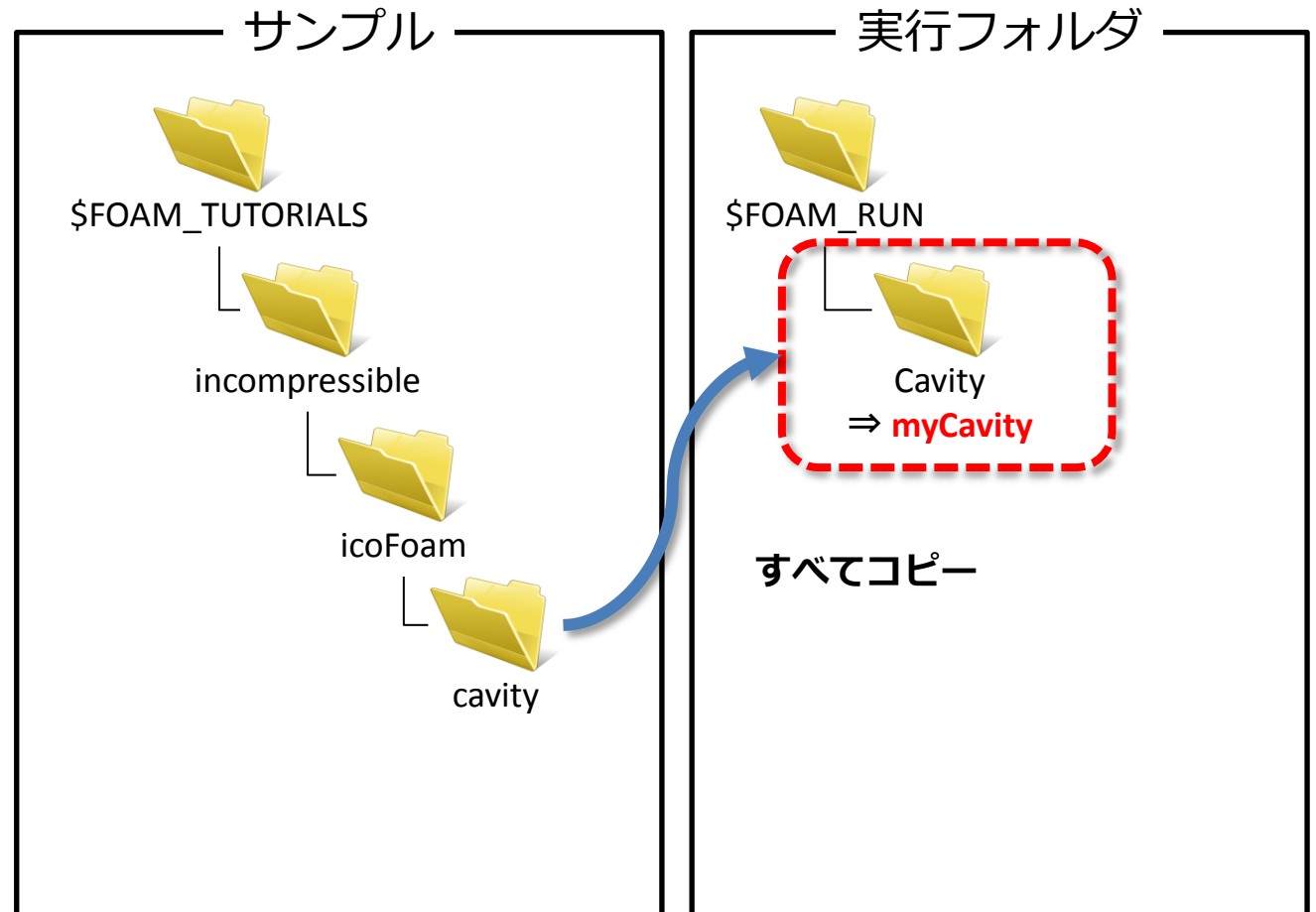
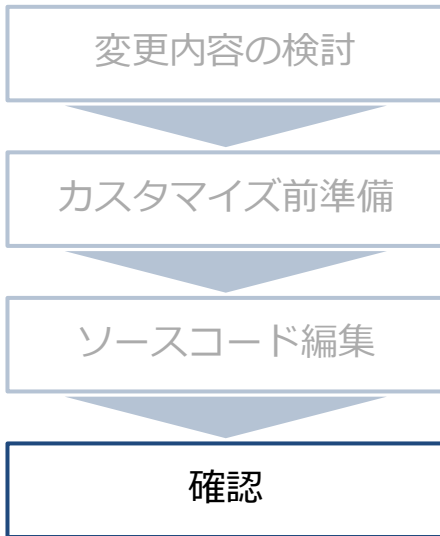
# まとめ

- Eclipseを利用したOpenFOAMカスタマイズを実施
- デバッグ / リリースの切替が非常に面倒
  - 環境変数の書き換え・読み直しが必要
  - ユニットテストはデバッグビルドで、システムテストでのデバッグはリリースビルドで実施で割り切るなら
- 質問
  - デバッグ / リリース切替の簡単なやり方ないですか？
  - OpenFOAMのデバッグライブラリは自分で作成するしかない？

# Appendix

# 確認

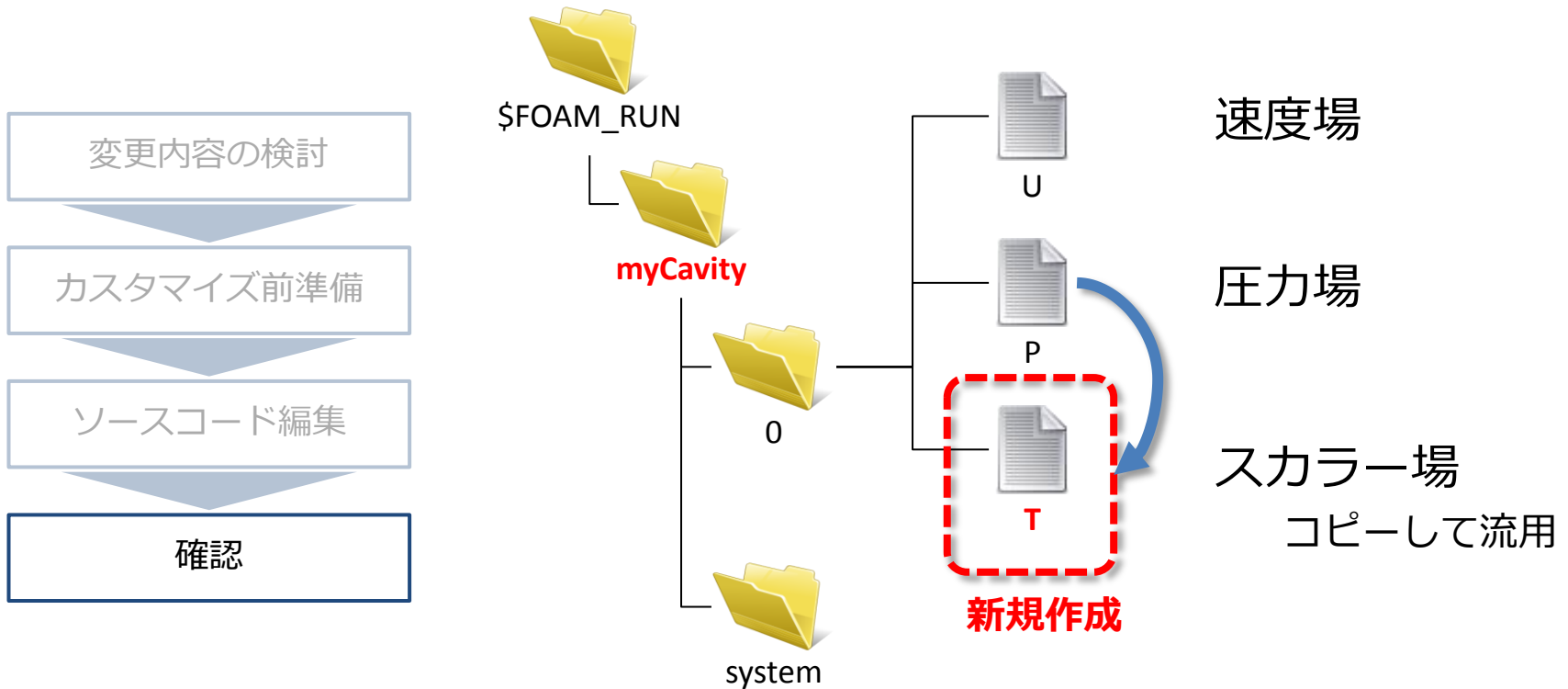
## ① ケースファイルの作成 - (1)



サンプルケースファイルから作成

# 確認

## ①ケースファイルの作成 - (2)



サンプルケースファイルから作成

# 確認

## ①ケースファイルの作成 – (3)



新規ファイル追加  
(pからコピー)

T



既存ファイル変更

system/fvSchemes

変更内容の検討

カスタマイズ前準備

ソースコード編集

確認

```
dimensions [0 0 0 0 0 0];
internalField uniform 0;
boundaryField
{
    movingWall
    {
        type fixedValue;
        value uniform 1;
    }
    fixedWalls
    {
        type fixedValue;
        value uniform 0;
    }
    frontAndBack
    {
        type empty;
    }
}
```

```
...
divSchemes
{
    default none;
    div(phi,U) Gauss linear;
    div(phi,T) Gauss linear;
}
laplacianSchemes
{
    default none;
    laplacian(nu,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(nu,T) Gauss linear corrected;
}
...
```

圧力場と同じ設定

# 確認

## ① ケースファイルの作成 - (4)



### 既存ファイル変更

system/fvSolution

```
...
solvers
{
  P
  { ...
  }
  U
  { ...
  }

  T
  {
    solver PBiCG;
    preconditioner DILU;
    tolerance 1e-05;
    relTol 0;
  }
}
...
```

圧力場と同じ設定

変更内容の検討

カスタマイズ前準備

ソースコード編集

確認

# 参考文献

- OpenFOAM on Eclipse, OpenCAE Gifu 第12回勉強会  
<http://opencae.gifu-nct.ac.jp/pukiwiki/index.php?%C2%E8%A3%B1%A3%B2%B2%F3%CA%D9%B6%AF%B2%F1%A1%A7H240310>
- Eclipse上でOpenFOAMを使う方法  
<http://mogura7.zenno.info/~et/xoops/modules/wordpress/index.php?p=443>
- HowTo Use OpenFOAM with Eclipse  
[http://openfoamwiki.net/index.php/HowTo\\_Use\\_OpenFOAM\\_with\\_Eclipse](http://openfoamwiki.net/index.php/HowTo_Use_OpenFOAM_with_Eclipse)