makePatchTypeField

At the end of `cylindricalInletVelocityFvPatchVectorField.C` we see

```
namespace Foam
{
    makePatchTypeField
    (
        fvPatchVectorField,
        cylindricalInletVelocityFvPatchVectorField
    );
}
```

This is a macro. A simple explanation is that whenever the *compiler* sees the macro `makePatchTypeField` with two parameters it will replace it with the definition of the macro.
It is a text-based replacement, so the parameters are here not formally classes.
The compiler will see this particular entry once this file is added to the compilation sequence.
The main purpose is to add the boundary condition to the run-time selection table.
It also sets the debug switch for the class.

Read more on macros at `https://en.cppreference.com/w/cpp/preprocessor/replace`

Let's try to figure out what the macro is replaced with for this particular class ...

makePatchTypeField

Find the definition of `makePatchTypeField` by

```
grep -r "#define makePatchTypeField" $FOAM_SRC
```

It is found in

```
$FOAM_SRC/finiteVolume/fields/fvPatchFields/fvPatchField/fvPatchField.H
```

In `fvPatchField.H`, we see

```
// For non-templated patch fields
#define makePatchTypeField(PatchTypeField, typePatchTypeField)                \
    defineTypeNameAndDebug(typePatchTypeField, 0);                            \
    addToPatchFieldRunTimeSelection(PatchTypeField, typePatchTypeField)
```

This means that the macro uses two other macros, to which it forwards the parameters.

Note that the `0` means that the debug switch is set to `0` by default! In the final code it is checked if the class itself sets another value.

Let's continue in the order that the macros are evaluated ...

defineTypeNameAndDebug

Find the definition of `defineTypeNameAndDebug` by

```
grep -r "#define defineTypeNameAndDebug" $FOAM_SRC
```

It is found in

```
$FOAM_SRC/OpenFOAM/db/typeInfo/className.H
```

In `className.H`, we see

```
//- Define the typeName and debug information
#define defineTypeNameAndDebug(Type, DebugSwitch)                       \
    defineTypeName(Type);                                               \
    defineDebugSwitch(Type, DebugSwitch)
```

Let's continue in the order that the macros are evaluated ...

defineTypeName and defineTypeNameWithName

Find the definition of `defineTypeName` by

```
grep -r "#define defineTypeName" $FOAM_SRC
```

It is found in

```
$FOAM_SRC/OpenFOAM/db/typeInfo/className.H
```

In `className.H`, we see

```
//- Define the typeName
#define defineTypeName(Type)                                                \
    defineTypeNameWithName(Type, Type::typeName_())
```

The definition of `defineTypeNameWithName` is also there:

```
//- Define the typeName, with alternative lookup as \a Name
#define defineTypeNameWithName(Type, Name)                                  \
    const ::Foam::word Type::typeName(Name)
```

Tracing back, we can see that:
`Type` is `cylindricalInletVelocityFvPatchVectorField`

Build-up of macro replacement (1/4)

The macro starts to be replaced by:

```
const ::Foam::word
    cylindricalInletVelocityFvPatchVectorField::typeName
    (
        cylindricalInletVelocityFvPatchVectorField::typeName_()
    );
```

Now we need to go back to where we latest branched off ...

defineDebugSwitch and defineDebugSwitchWithName

Find the definition of `defineDebugSwitch` by

```
grep -r "#define defineDebugSwitch" $FOAM_SRC
```

It is found in

```
$FOAM_SRC/OpenFOAM/global/debug/defineDebugSwitch.H
```

In `defineDebugSwitch.H`, we see

```
//- Define the debug information
#define defineDebugSwitch(Type, Value)                                      \
    defineDebugSwitchWithName(Type, Type::typeName_(), Value);              \
    registerDebugSwitchWithName(Type, Type, Type::typeName_())
```

The definition of `defineDebugSwitchWithName` is also there:

```
//- Define the debug information, lookup as \a Name
#define defineDebugSwitchWithName(Type, Name, Value)                        \
    int Type::debug(::Foam::debug::debugSwitch(Name, Value))
```

Build-up of macro replacement (2/4)

The macro continues to be replaced by:

```
int cylindricalInletVelocityFvPatchVectorField::debug
    (
        ::Foam::debug::debugSwitch
            (
             cylindricalInletVelocityFvPatchVectorField::typeName_(), 0
            )
    );
```

Now we need to go back to where we latest branched off ...

registerDebugSwitchWithName

Find the definition of `registerDebugSwitchWithName` by

```
grep -r "#define registerDebugSwitchWithName" $FOAM_SRC
```

It is found in

`$FOAM_SRC/OpenFOAM/global/debug/defineDebugSwitch.H`

In `defineDebugSwitch.H`, we see

```
//- Define the debug information, lookup as \a Name
#define registerDebugSwitchWithName(Type,Tag,Name)                              \
    class add##Tag##ToDebug                                                     \
    :                                                                           \
        public ::Foam::simpleRegIOobject                                        \
    {                                                                           \
    public:                                                                     \
                                                                                \
        explicit add##Tag##ToDebug(const char* name)                            \
        :                                                                       \
            ::Foam::simpleRegIOobject(::Foam::debug::addDebugObject, name)      \
        {}                                                                      \
                                                                                \
        virtual ~add##Tag##ToDebug() = default;                                 \
                                                                                \
        virtual void readData(::Foam::Istream& is)                              \
        {                                                                       \
            is >> Type::debug;                                                  \
        }                                                                       \
                                                                                \
        virtual void writeData(::Foam::Ostream& os) const                       \
        {                                                                       \
            os << Type::debug;                                                  \
        }                                                                       \
                                                                                \
        add##Tag##ToDebug(const add##Tag##ToDebug&) = delete;                   \
        void operator=(const add##Tag##ToDebug&) = delete;                      \
    };                                                                          \
    add##Tag##ToDebug add##Tag##ToDebug_(Name)
```

registerDebugSwitchWithName

In the previous slide we could see entries containing:

`##Tag##`

This concatenates the content of `Tag` with the text before and after. In the case of

`add##Tag##ToDebug`

where `Tag` here has the value `cylindricalInletVelocityFvPatchVectorField`, we get:

`addcylindricalInletVelocityFvPatchVectorFieldToDebug`

Build-up of macro replacement (3/4)

The macro continues to be replaced by:

```
class addcylindricalInletVelocityFvPatchVectorFieldToDebug
:
    public ::Foam::simpleRegIOobject
{
public:

    explicit addcylindricalInletVelocityFvPatchVectorFieldToDebug(const char* name)
    :
        ::Foam::simpleRegIOobject(::Foam::debug::addDebugObject, name)
    {}

    virtual ~addcylindricalInletVelocityFvPatchVectorFieldToDebug() = default;

    virtual void readData(::Foam::Istream& is)
    {
        is >> cylindricalInletVelocityFvPatchVectorField::debug;
    }

    virtual void writeData(::Foam::Ostream& os) const
    {
        os << cylindricalInletVelocityFvPatchVectorField::debug;
    }

    addcylindricalInletVelocityFvPatchVectorFieldToDebug(const addcylindricalInletVelocityFvPatchVectorFieldToDebug&) = delete;
    void operator=(const addcylindricalInletVelocityFvPatchVectorFieldToDebug&) = delete;
};
addcylindricalInletVelocityFvPatchVectorFieldToDebug addcylindricalInletVelocityFvPatchVectorFieldToDebug_(cylindricalInletVelocityFvPatchVectorField::typeName_());
```

Now we need to go back to where we latest branched off ...

addToPatchFieldRunTimeSelection

Find the definition of `addToPatchFieldRunTimeSelection` by

```
grep -r "#define addToPatchFieldRunTimeSelection" $FOAM_SRC
```

It is found in

`$FOAM_SRC/finiteVolume/fields/fvPatchFields/fvPatchField/fvPatchField.H`

In `fvPatchField.H`, we see

```
#define addToPatchFieldRunTimeSelection(PatchTypeField, typePatchTypeField)     \
    addToRunTimeSelectionTable                                                  \
    (                                                                           \
        PatchTypeField,                                                         \
        typePatchTypeField,                                                     \
        patch                                                                   \
    );                                                                          \
    addToRunTimeSelectionTable                                                  \
    (                                                                           \
        PatchTypeField,                                                         \
        typePatchTypeField,                                                     \
        patchMapper                                                             \
    );                                                                          \
    addToRunTimeSelectionTable                                                  \
    (                                                                           \
        PatchTypeField,                                                         \
        typePatchTypeField,                                                     \
        dictionary                                                              \
    );
```

addToRunTimeSelectionTable

Find the definition of `addToRunTimeSelectionTable` by

```
grep -r "#define addToRunTimeSelectionTable" $FOAM_SRC
```

It is found in

```
$FOAM_SRC/OpenFOAM/db/runTimeSelection/construction/addToRunTimeSelectionTable.H
```

In `addToRunTimeSelectionTable.H`, we see

```
//- Add to hash-table of functions with typename as the key
#define addToRunTimeSelectionTable\
(baseType,thisType,argNames)                                                \
                                                                            \
    /* Add the thisType constructor function to the table */                \
    baseType::add##argNames##ConstructorToTable<thisType>                   \
        add##thisType##argNames##ConstructorTo##baseType##Table_
```

Build-up of macro replacement (4/4)

The macro continues to be replaced by:

```
/*Add the cylindricalInletVelocityFvPatchVectorField constructor function to the table*/
fvPatchVectorField::addpatchConstructorToTable<cylindricalInletVelocityFvPatchVectorField>
addcylindricalInletVelocityFvPatchVectorFieldpatchConstructorTofvPatchVectorFieldTable_;

/*Add the cylindricalInletVelocityFvPatchVectorField constructor function to the table*/
fvPatchVectorField::addpatchMapperConstructorToTable<cylindricalInletVelocityFvPatchVectorField>
addcylindricalInletVelocityFvPatchVectorFieldpatchMapperConstructorTofvPatchVectorFieldTable_;

/*Add the cylindricalInletVelocityFvPatchVectorField constructor function to the table*/
fvPatchVectorField::adddictionaryConstructorToTable<cylindricalInletVelocityFvPatchVectorField>
addcylindricalInletVelocityFvPatchVectorFielddictionaryConstructorTofvPatchVectorFieldTable_;
```

Now we are done! This is what the compiler does before compiling the code.

Manually replace the macro

We can test that this description is correct by replacing the macro with the pieces of code stated in the slides "Build-up of macro replacement".

Copy and compile the `cylindricalInletVelocity` boundary condition and name it `myCylindricalInletVelocity`, as we have previously done.

Replace the macro in `myCylindricalInletVelocityFvPatchVectorField.C`:

```
namespace Foam
{
/*
    makePatchTypeField
    (
        fvPatchVectorField,
        myCylindricalInletVelocityFvPatchVectorField
    );
*/
<CODE FROM SLIDES "Build-up of macro replacement", with myCyl... instead of cyl...>
}
```

Compile, set up an `icoFoam/cavity` case with the boundary condition as we did before, and make sure that you get the same results as always.