Aalto University
School of Engineering

Exercise 4.2 – Modeling a Parametric LEGO Brick

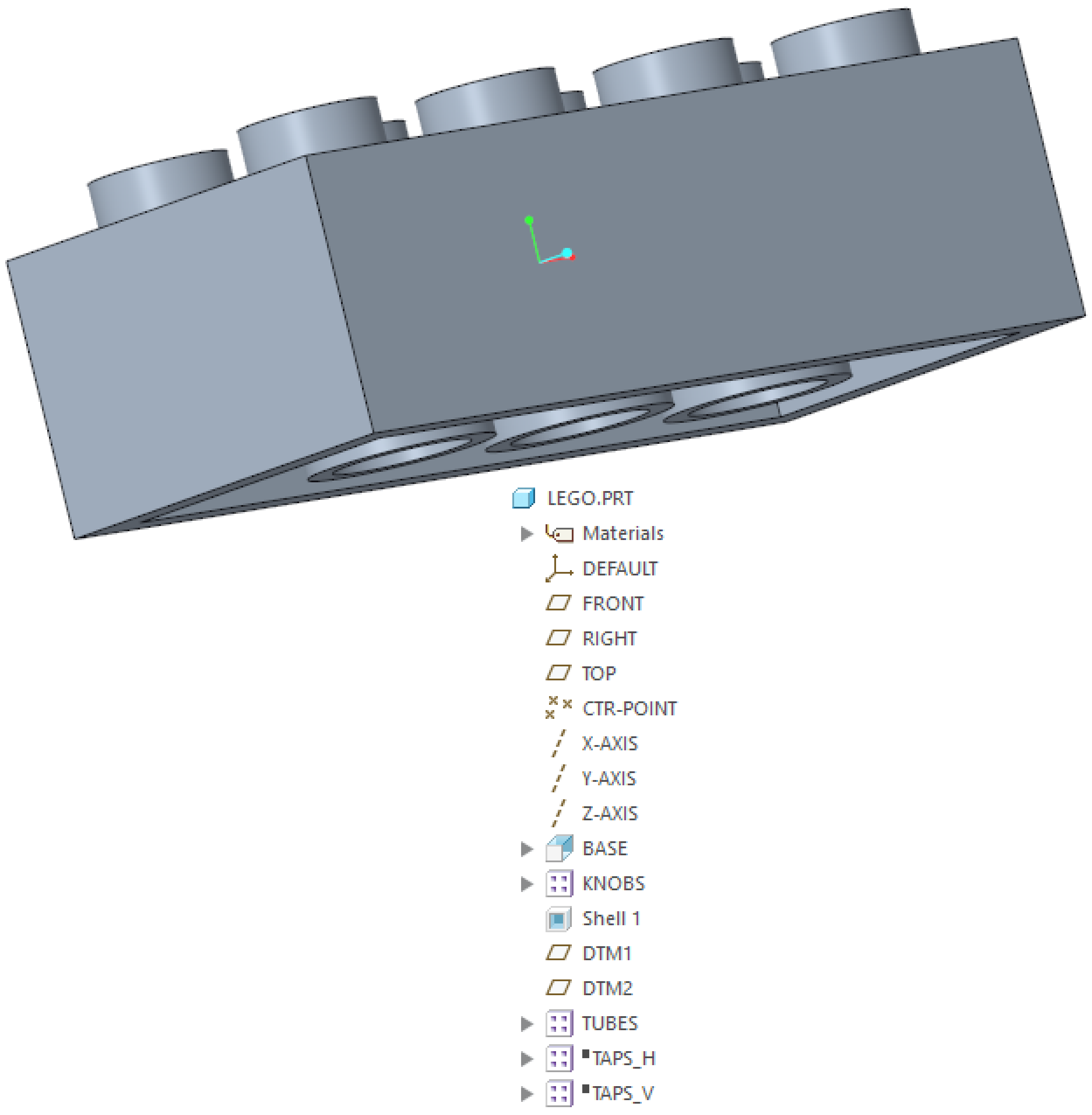


Figure 1: Lego brick and its model tree.

Learning Targets

In this exercise you will learn:

- ✓ Using Relations
- ✓ Using Parameters
- ✓ Suppressing features through Pro/Program
- ✓ Asking inputs using Pro/Program

It is assumed, that all previous exercises are done. The scope of this exercise is in the field of relations, parameters and program, not in creating basic features.

About this Exercise

This exercise is an example of what can be achieved by using relations-programming and Pro/Program-scripts in collaboration. The result is a highly parametric Lego block part model, which has all the required intelligence built-in in order to create all possible basic block types by just adjusting a few simple parameters.

Parameters

Parameters are stored variables inside a part file, which are created by the user. These can be of integer, real number or text type, and they can be used for example relations-scripting and passing information between the model and the Pro/Program-module. Parameters can be created and edited from *Model Intent* group in *Model* or *Tools* tab.

Relations

Relations-script is a simple programming script that is run at least once every time the model is regenerated. Commands in the relations-script can be used to control model dimensions and parameters in a programming-language-like environment. This enables more complex dependencies between parameters and dimensions than traditional referencing and constraining. Relations can live on three different levels from top to bottom:

1) Assembly relations, which can have control over all dimensions inside an assembly

2) Part relations, which can control all dimensions of the current part

3) Sketch relations, which can control the dimensions of one sketch. This relations-script is calculated real-time as the sketch is being extruded (for example as a sketch inside a variable section sweep) allowing for complex changes during the sweep action.

By default, models have no relations script code, but all three levels can be made exist at the same time. The relations creator/editor can be found from *Model Intent* group in *Model* or *Tools* tab.

Pro/Program

The Pro/Program-script is also a programming-language-like script that is run exactly once per every regeneration cycle. All assembly and part files do by default have this kind of script initiated with some actions. Its purpose is to coordinate the creation of the model, which includes adding features, parts and possibly the relations-listings. Program-script can be modified for example to control the presence of certain features or parts according to parameter values. The program editor can be found from *Model Intent* group in *Model* or *Tools* tab.

About Basic Lego Brick

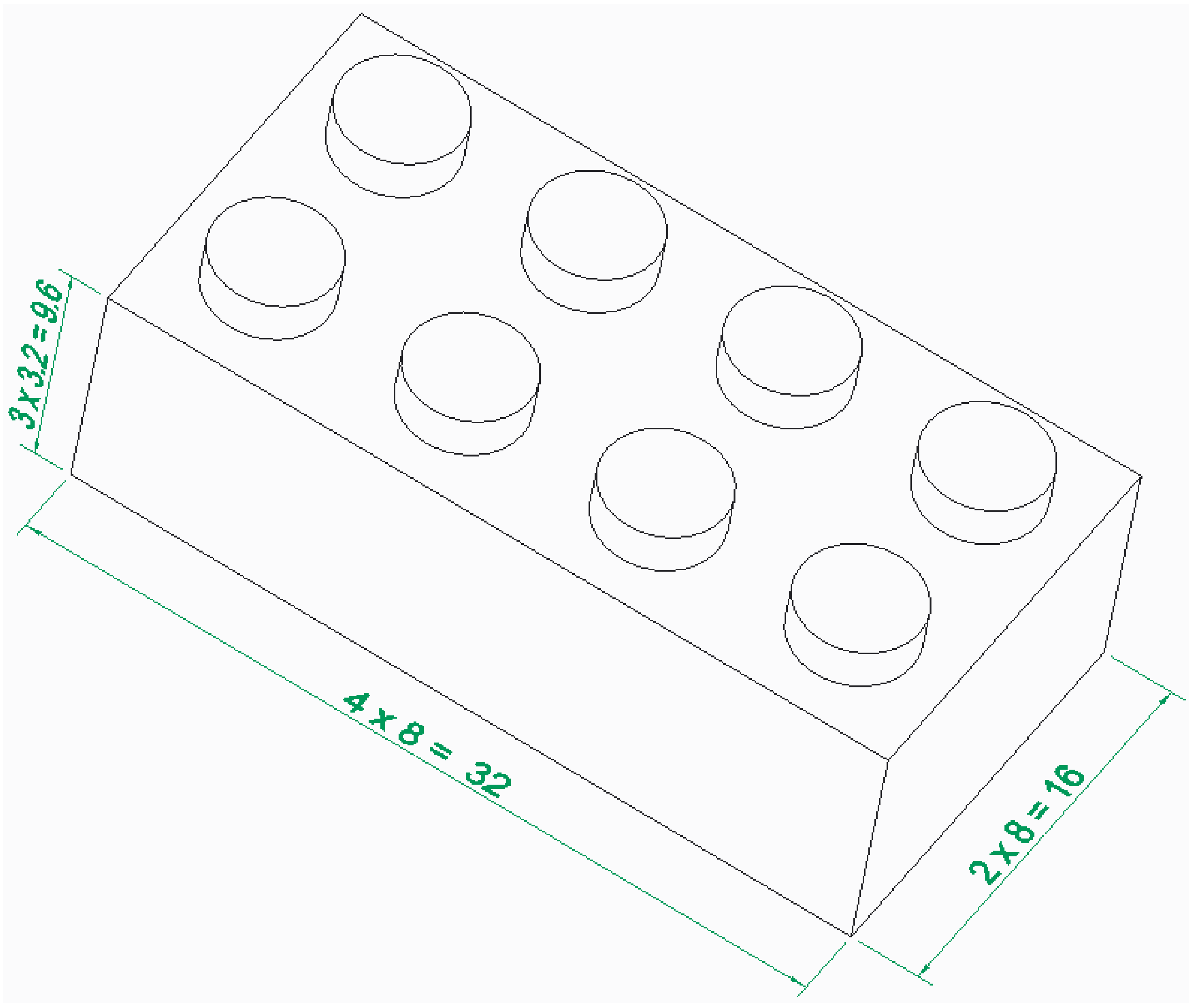


Figure 2: Basic 4 x 2 Lego brick and its dimensions.

The most common and well-known Lego brick is presented in Figure 2. The picture shows a 4 x 2 Lego brick: the 4 is the length of the brick and 2 is the width of the brick. The brick is build using eight same sized modules. One basic module is shown in Figure 3. By multiplying this module, all basic bricks are created. This 8 is a module for length and width.

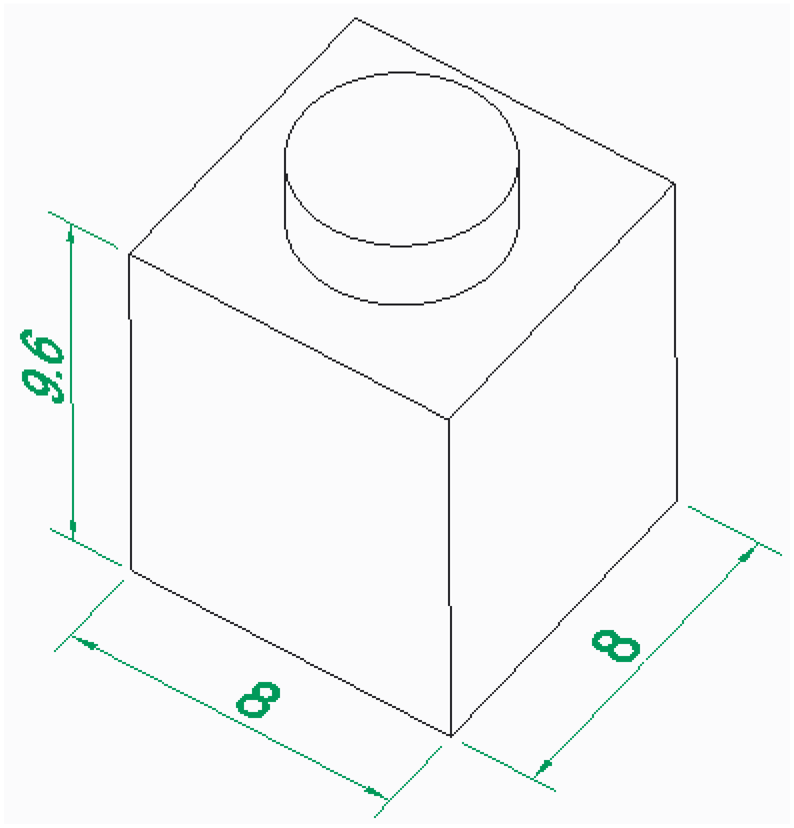


Figure 3: One basic Lego brick module.

The module for the height is 3.2. Basic Lego brick has the height of 9.6, which is 3 times 3.2. However, unlike with length and width, the height is fixed; it can be normal (9.6) or thin (3.2).

Lego brick has knobs on the top to allow attachment to other bricks. The diameter of the knob is 4.8, the height of the knob is 1.8 and it is located in the middle (8 / 2 = 4) of the basic module (Figure 4).

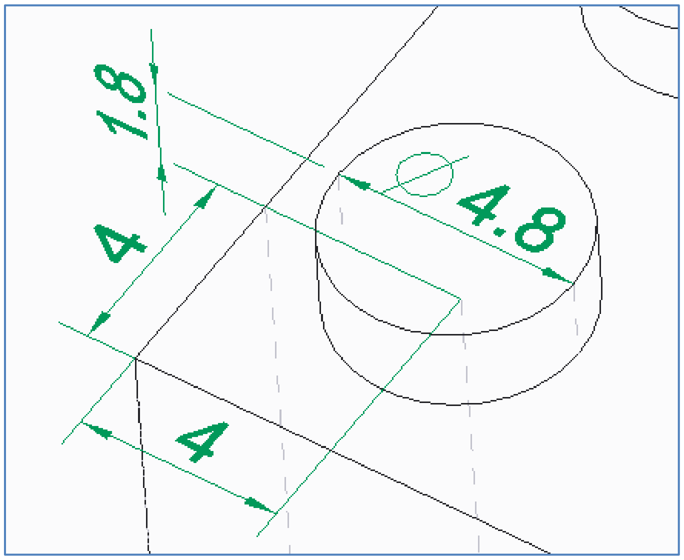


Figure 4: Lego brick knob.

To make Lego bricks connected with each other, the brick has tubes at the bottom of the hollow brick. The outer diameter of the tube is 6.5 and inner 4.8 (the same as the diameter of the taps!). The height of the tube is the height of the brick - 1; for normal brick it is 8.6, for thick brick 2.2. The tube is located in the middle of four basic 8 modules (Figure 5).

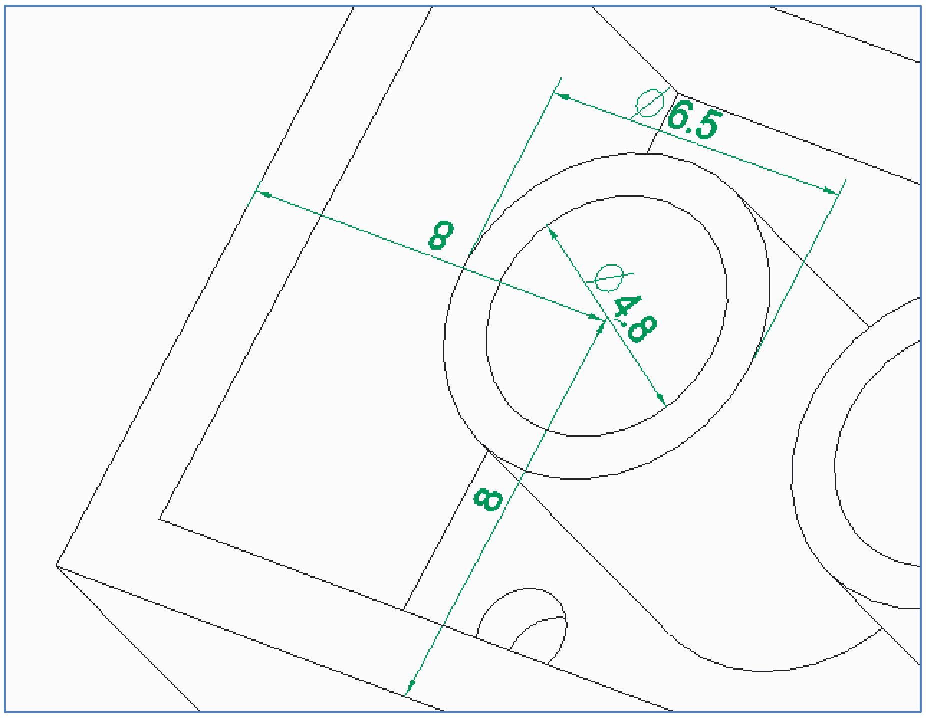


Figure 5: The bottom tube of the Lego brick.

There are also taps for the hollow bottom and those are only present when the width of the brick is 1. In this situation, the tubes are removed. The height of the tap is the same as the tube's, the diameter is 3.2 and it is located in the middle of two basic 8 modules (Figure 6).

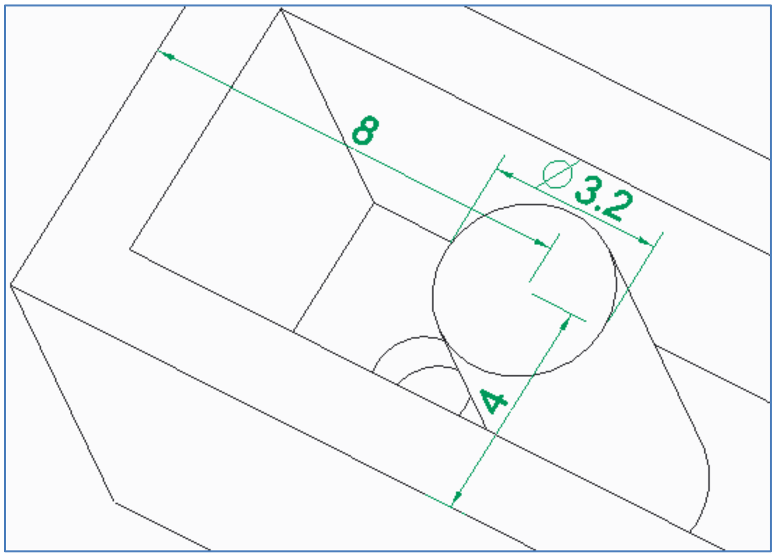


Figure 6: The Lego bricks tap (only when width or length is 1).

As you have noticed from previous pictures, the Lego brick is hollow. The wall thickness on the sides are 1.6 and in the top 1 (Figure 7). There is other configuration of the brick where wall thickness is 1.3 and the needed tightness is created by supporting ribs (thickness of 0.3). In our exercise, we use the version without the ribs.

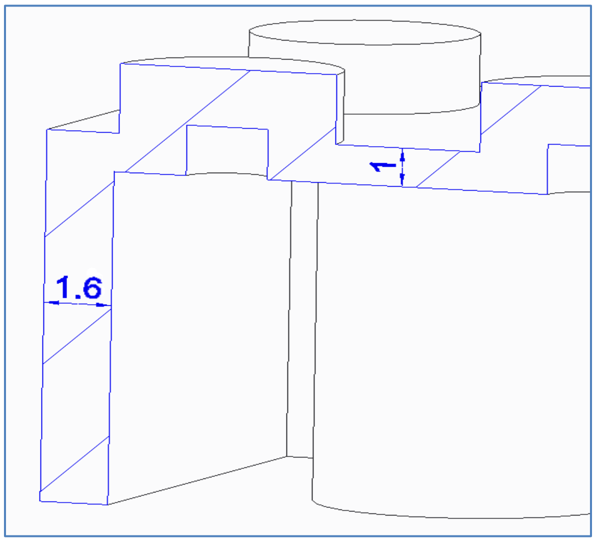


Figure 7: The thicknesses of the Lego brick.

Getting started

Create a **new solid part** model and name it as LEGO.

The first thing about modeling is to decide the “layout”, how the model will be placed with respect to basic datum geometry, for instance. Figure 8 shows the basic layout used in this exercise. The TOP datum plane will define the bottom of the block, while RIGHT and FRONT planes will coincide in the middle of the first knob. This will be the knob and thus module that is always present.

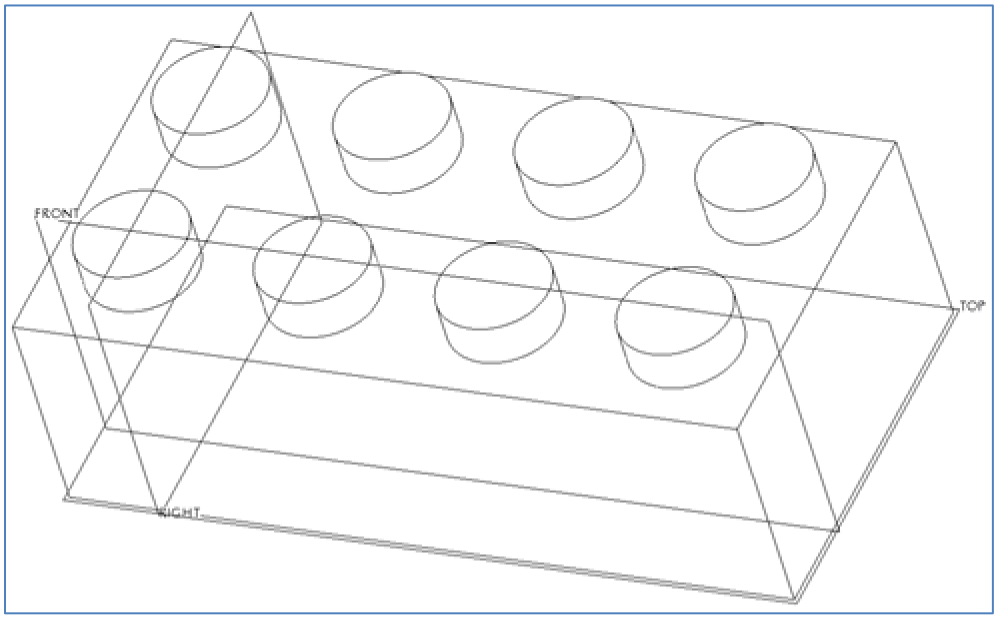


Figure 8: Layout of the basic 4 x 2 Lego brick and part's basic datum planes.

First Features

Base geometry

Using **Extrude**, sketch on the TOP plane as shown in Figure 9, using the other planes as references (*Reference* plane RIGHT and *Orientation Right*). Extrude blind with dimension of **9.6**. Rename extrude as BASE.

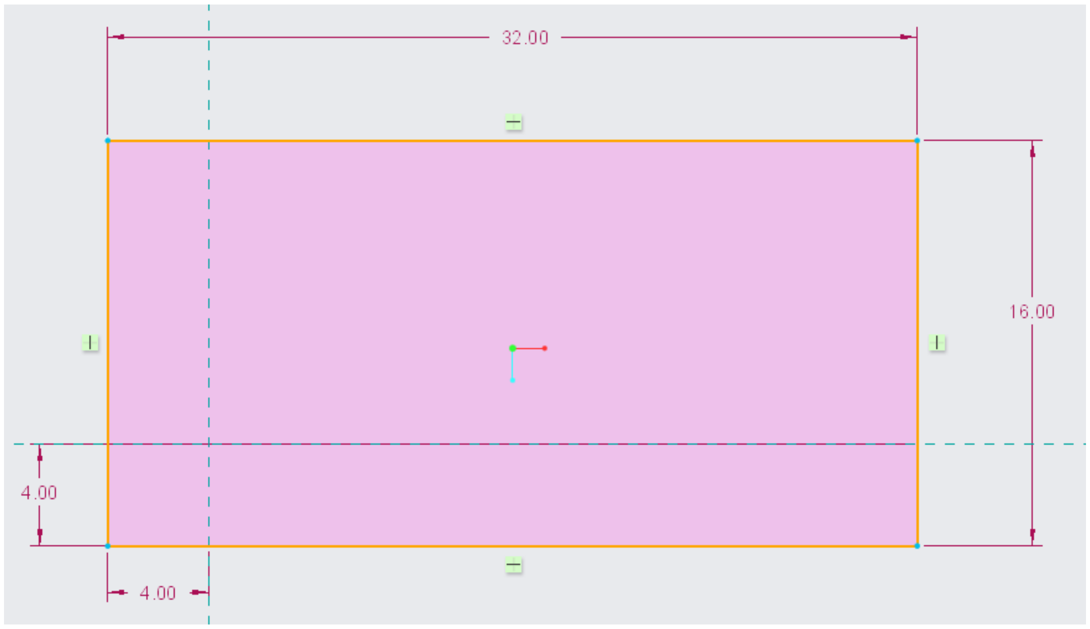


Figure 9: Sketch for the first feature (BASE).

Knob

Create an **extrude** on the upper face of the BASE feature to make the first knob. Using RIGHT and FRONT as references, sketch a circle with the diameter of **4.8**. Extrude it blindly by **1.8** (Figure 10). Rename to feature to KNOB.

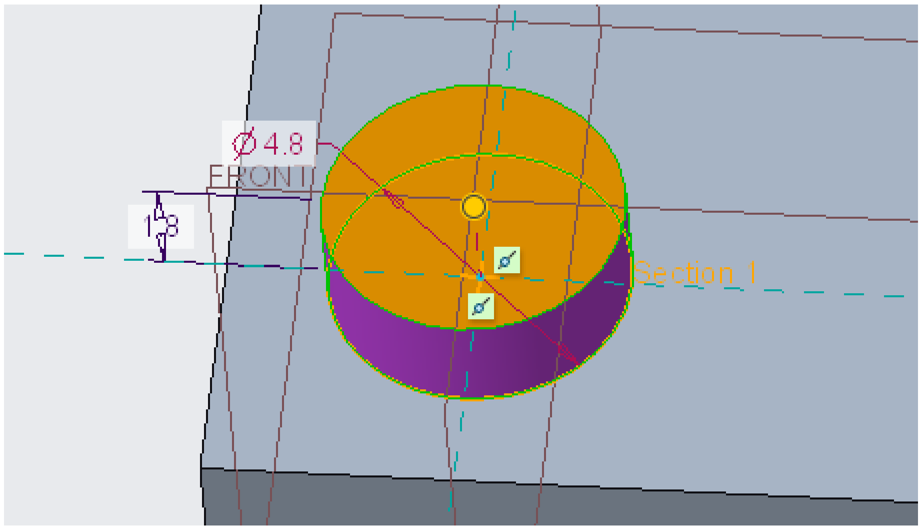
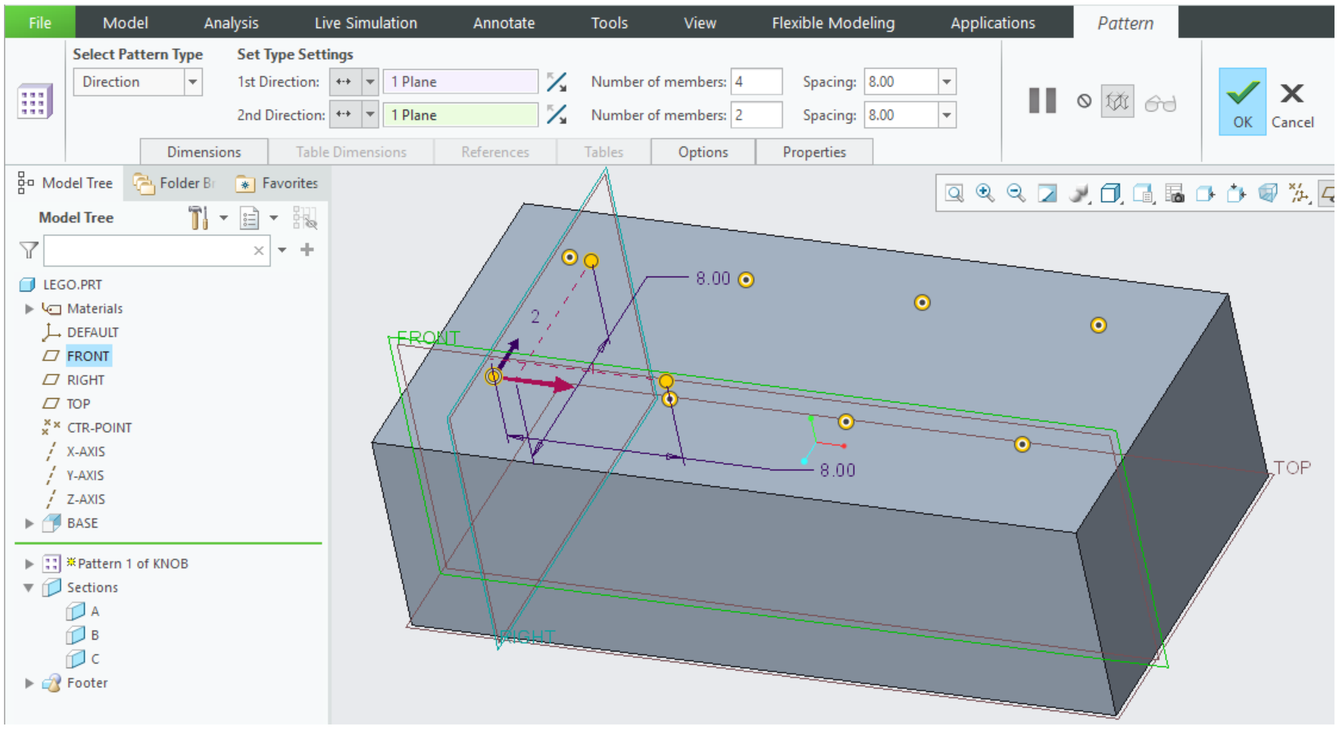


Figure 10: The first knob and its dimensions.

Pattern of Knobs

Pattern the KNOB using the **Direction** method and two direction references, RIGHT and FRONT planes. With **8** increments and sufficient numbers of instances (**4** and **2**), create the needed knobs (Figure 11). Rename the created pattern feature as KNOBS.



Select Pattern Type: Direction
Set Type Settings
1st Direction: 1 Plane — Number of members: 4 — Spacing: 8.00
2nd Direction: 1 Plane — Number of members: 2 — Spacing: 8.00

Figure 11: Patterning the KNOB. Notice the 1st and 2nd directions.

Shell

The Lego brick is hollow, but the sides and the top have different thicknesses. Using the **Shell** (, *Engineering* group) tool, create first the shell with the thickness value of **1.6** and remove the bottom surface. Then in the **References** tab, there is a field for *Non-default thicknesses*. Click that field and choose the upper surface of the BASE feature and give a new value of **1** (Figure 12).

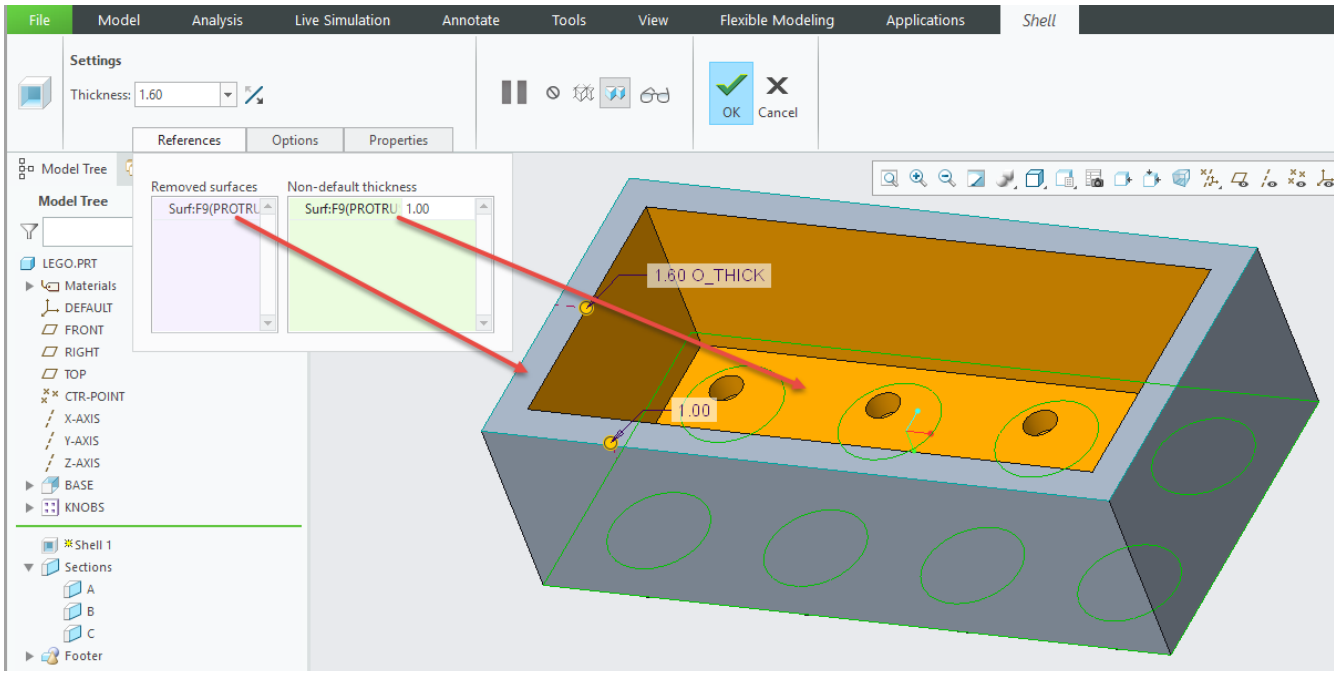


Figure 12: Ready to accept Shell feature. The non-default thickness surface is highlighted in green.

Creating Parameters

Now it is the time to create the first parameters, followed by some relations between them and the dimensions of the model. Select **Parameters** ([ ]) from the *Model Intent* group. A *Parameters* window appears. If there are already some parameters, do not care about them. Use the plus sign (+) to add three parameters (Table 1).

Table 1: Created parameters and their default values.

Name	Type	Value
Size_h	Integer	4
Size_v	Integer	2
Thin	Yes No	No

Creo is not case sensitive. The size parameters refer to numbers of knobs, thin determines if block height is 9.6 (no) or 3.2 (yes). Click **OK**.

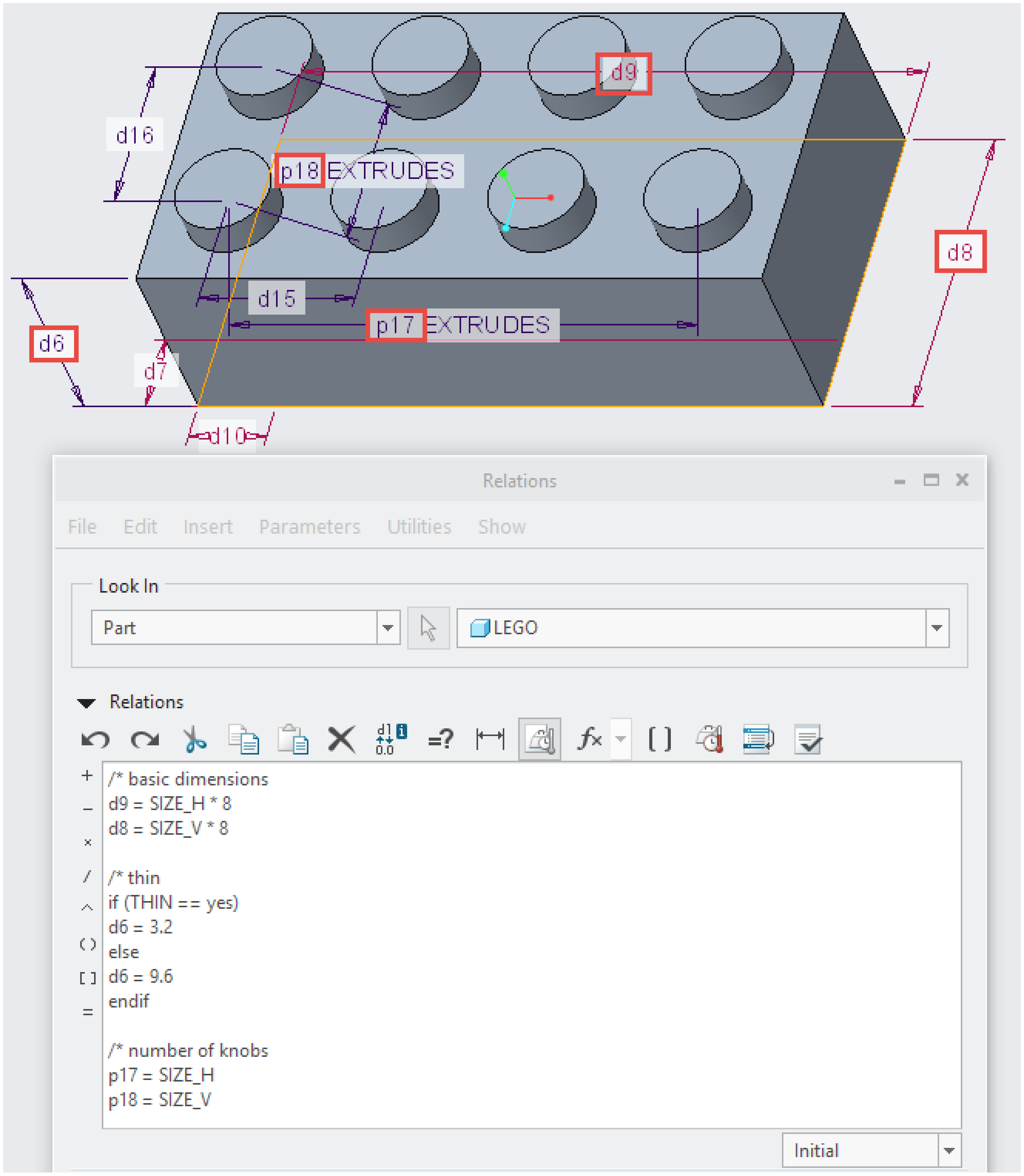
Creating Relations

To make these parameters affect to the dimensions of the model, some relations will be added next. Select **Relations** (d=) from *Model Intent* group. Write in the *Relations* field (see also Figure 13) as seen in Table 2.

Table 2: Created Relations code.

```
/* basic dimensions
d9 = SIZE_H * 8
d8 = SIZE_V * 8

/* thin
if (THIN == yes)
d6 = 3.2
else
d6 = 9.6
endif

/* number of knobs
p17 = SIZE_H
d16 = SIZE_V
```

d# are dimensions symbols, p# are symbols for numbers of pattern instances. When *Relations* editor is active, you can click features in the model tree or on the graphics window to show respective symbols. Use **CTRL+R** or **Repaint** () from *Graphics* toolbar to clear them if needed. The symbols can be copied into the relations editor by clicking on them or writing them manually. Replace the symbols in the **boldface** font in Table 2 with appropriate ones from your model. Please notice that because Creo names dimensions according to the creation method and order, your dimensions may have different names! To check that program understands your code, click the **Verify** () button on the *Relations* window.



```
/* basic dimensions
d9 = SIZE_H * 8
d8 = SIZE_V * 8

/* thin
if (THIN == yes)
d6 = 3.2
else
d6 = 9.6
endif

/* number of knobs
p17 = SIZE_H
p18 = SIZE_V
```

Figure 13: Relations window and part's dimensions. Notice, that you may have different names for the dimensions!

The lines starting with /* are comments, Creo ignores them. Those are used to clarify the code. The line d9 = size_h * 8 means that the value of dimension d3 is set to the value of parameter size_h multiplied by eight. That is, the value on the right side of "=" is always assigned to the left side.

The if – else – endif block is used to check which of two possible values parameter thin has, and then assign proper value to dimension d6 (the depth of the first extrusion). Note that "==" is logical operator ("equals"), whereas "=" is for value assignment.

When ready, accept the written relations by selecting **OK**. Check that the model works: change parameter values (**Parameters** ([ ]) from *Model Intent* group) and regenerate the model (**CTRL+G**). After some parameter testing, change *Size_h* to **4**, *Size_v* to **2** and *thin* to **NO**.

Remember to save your model!

Creating More Features

Creating planes

Create a **datum plane** (▱, *Datum* group) using the default offset method, **4** as offset value and RIGHT plane as reference. Repeat with FRONT plane. See Figure 14.

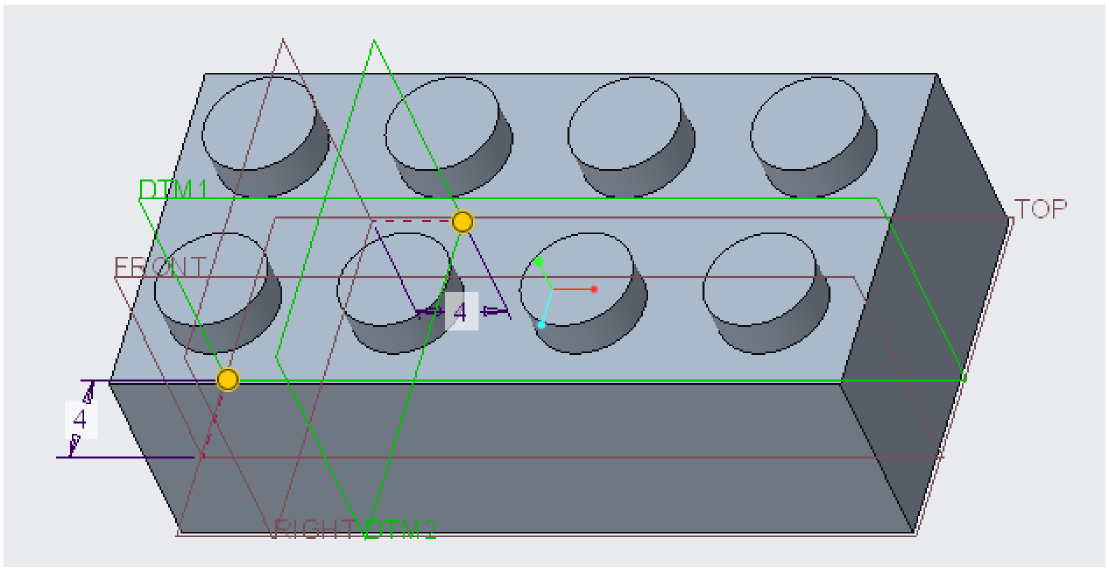


Figure 14: Two newly created planes highlighted.

Tube

Using **Extrude**, sketch on TOP plane two concentric circles with the diameters of **4.8** and **6.5**, using the newly created datum planes (DTM1 and DTM2) as placement references (Figure 15). Use of the datum planes has actually cosmetic nature only, RIGHT and FRONT planes and suitable dimensions (4 and 4) could have been used in the sketch as well. Extrude upwards using **To Next** (☰) option to create a tube. Name the created feature to TUBE.

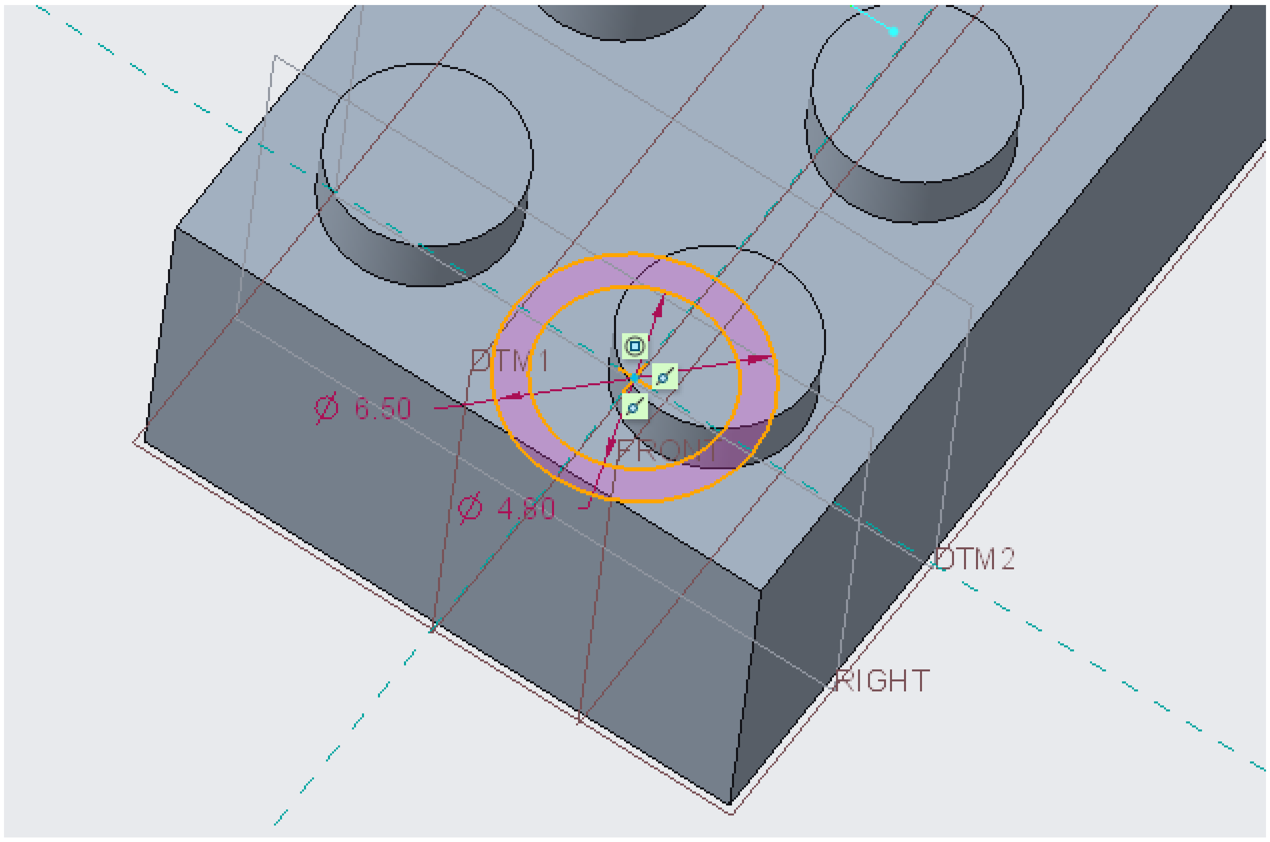


Figure 15: Sketched two concentric circles. Planes DTM1 and DTM2 used as references.

Patterning the Tube

Pattern the tube using the direction method, the new datum planes (DTM2 for 1st and DTM1 for 2nd direction) as references, **8** as distance and **3** and **1** as numbers of instances (Figure 16). To keep the model tree easy to read, rename pattern to TUBES. Save the model.

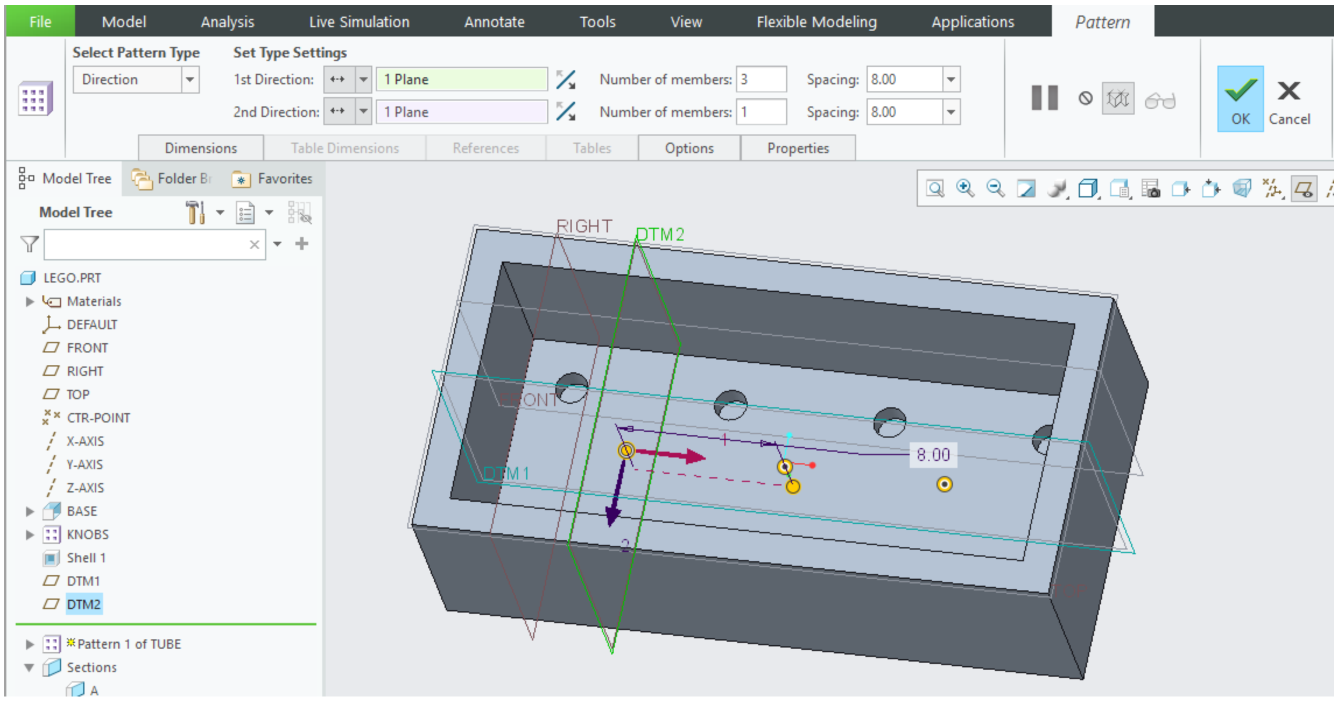


Figure 16: Ready to accept pattern to pattern TUBE.

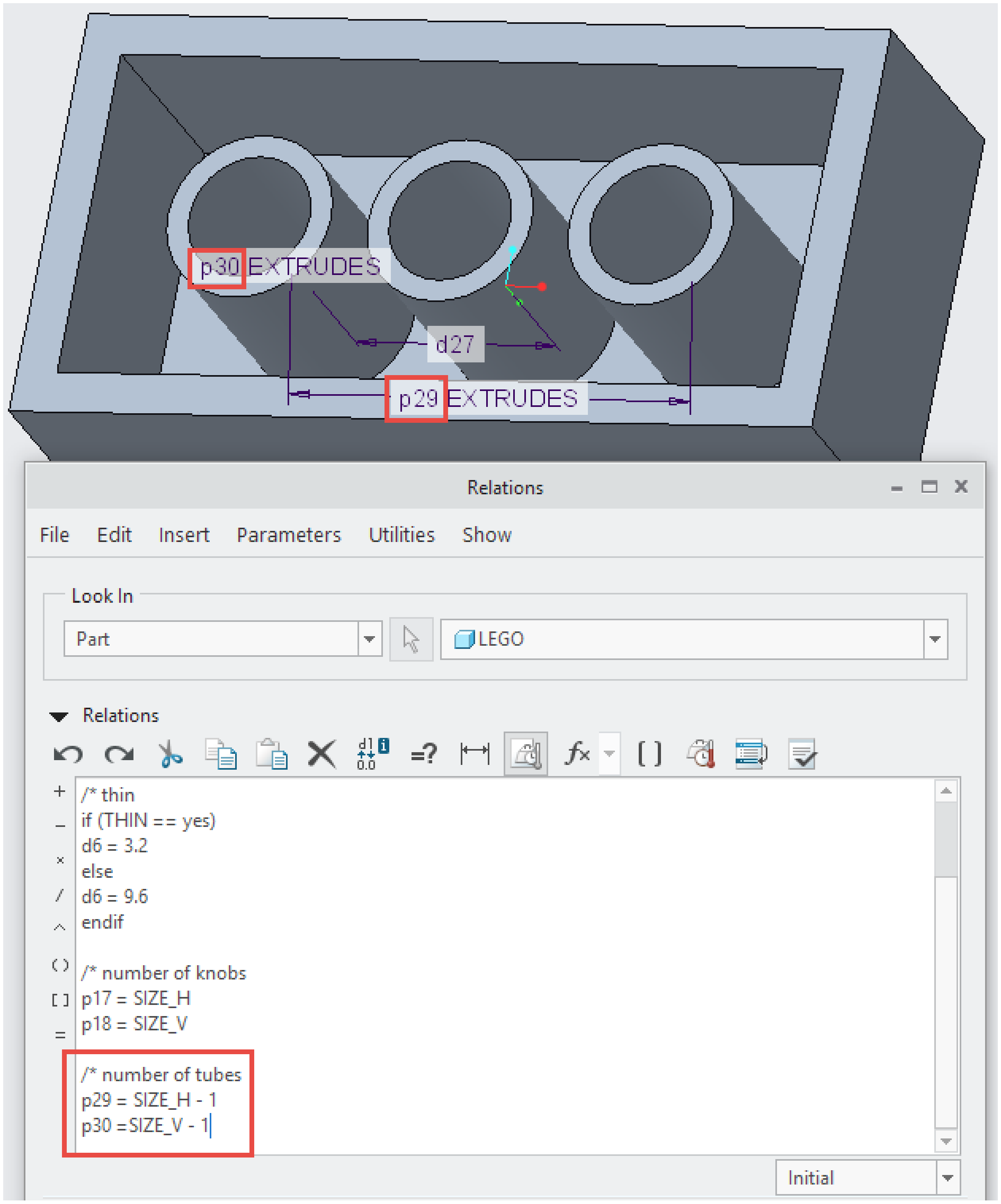
Updating Relations

Select **Relations** (d=) from *Model Intent* group and write the following new lines to the end (Table 3):

Table 3: New lines added to the end of the relations.

```
/* number of tubes
p29 = size_h - 1
p30 = size_v - 1
```

Replace p29 and p30 with appropriate symbols from your model (Figure 17). Set *Size_h* and *Size_v* to **2** and regenerate the model. Having only one tube facilitates the following steps. Do not try other values at this point!



```
/* thin
if (THIN == yes)
d6 = 3.2
else
d6 = 9.6
endif

/* number of knobs
p17 = SIZE_H
p18 = SIZE_V

/* number of tubes
p29 = SIZE_H - 1
p30 =SIZE_V - 1
```

Figure 17: Relations window and new lines added to the end.

Suppressing Features

Unfortunately, the number of pattern instances cannot be set to zero. That is, the current model will not work with value 1 of *size_h* or *size_v*. Instead of setting the number of pattern instances to zero, the pattern has to be suppressed. Suppressing means temporarily deleting, so that suppressed features can easily be restored (by resume command). Besides manually, features can be suppressed using Pro/Program.

Changes in the relations

Before using Pro/Program, create a **new** YES NO -type **parameter** called *tubes*. A few modifications must be made to relations as well (Table 4).

Table 4: Modified relations. The lines in *italic* were added in previous chapter.

```
/* number of tubes
if (size_h > 1 & size_v > 1)
tubes = yes
p29 = size_h - 1
p30 = size_v - 1
else
tubes = no
p29 = 1
p30 = 1
endif
```

Again, p29 and p30 naturally must be replaced with proper symbols. Click **OK** when ready. At this point, next to nothing has been done to actually suppress the tube pattern; using the help parameter *tubes* is even not necessary, but it makes the model easier to use.

Pro/Program

Select **Program** from the *Model Intent* group and select **Edit Design** from the *Menu Manager*. The program of the model is opened by the default text editor, in Aalto class case by Notepad. The Pro/Program is instruction for the program to build the model; it shows what features and what values the features have. The structure of Pro/Program is showed in Table 5.

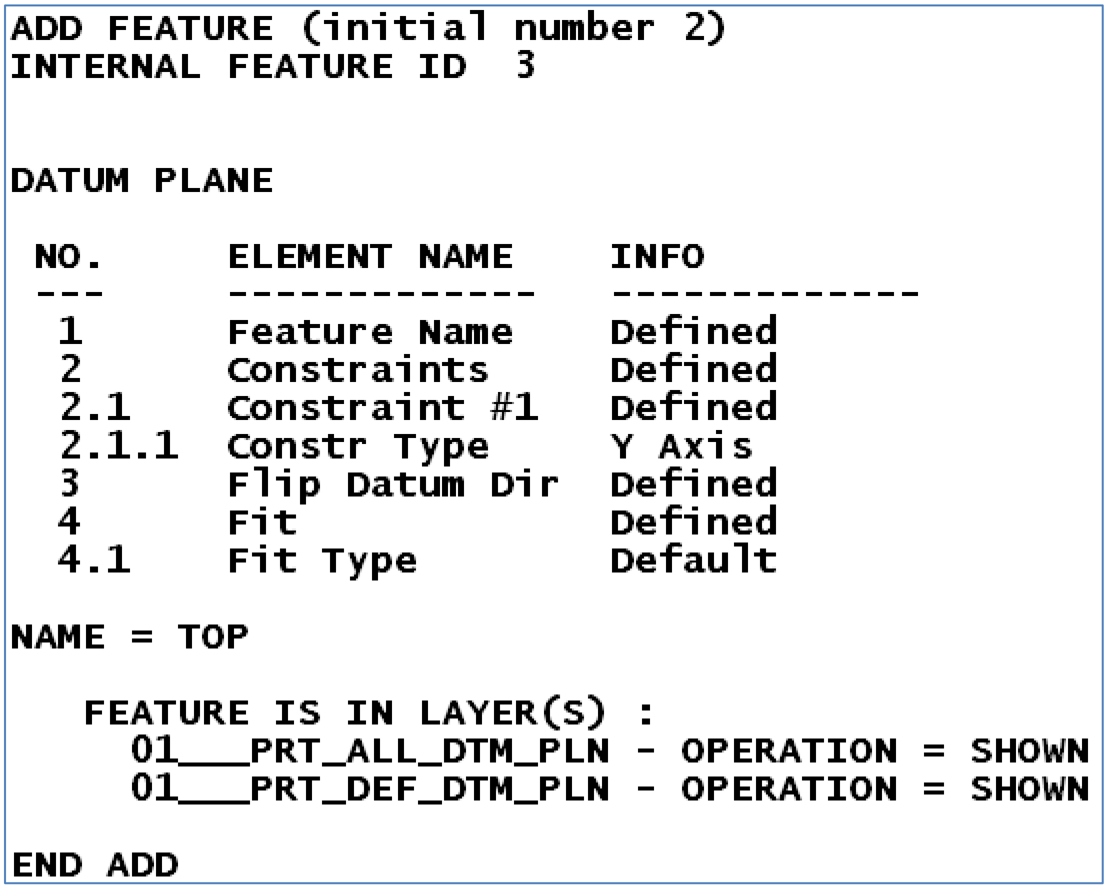
Table 5: Structure of the Pro/Program.

```
[version info & model name]

INPUT
END INPUT

RELATIONS
[relations from the Relations window, i.e. what user has made]
END RELATIONS

ADD FEATURE (initial number 1)
[feature information and properties]
END ADD

…

ADD FEATURE (initial number N)
[feature information and properties]
END ADD

MASSPROP
END MASSPROP
```

When looking the Pro/Program the first time, it may look fuzzy, but it has a proper structure. In the beginning there is version information (three lines), then there are all inputs to the program (in our case this field is empty, yet), then all relations made at the part level (Relations window), then all features the model have in the same order as in the *Model Tree* and in the end mass properties field.

Feature

The Pro/program lines to define the first datum plane feature are showed in Figure 18. As you can notice, the block to add one feature to the model starts with ADD FEATURE and ends with END ADD. The type of the feature is listed (DATUM PLANE) and the name of the feature also (NAME = TOP). That's all we need to know about one single feature.



```
ADD FEATURE (initial number 2)
INTERNAL FEATURE ID  3


DATUM PLANE

 NO.      ELEMENT NAME    INFO
 ---      -------------   -------------
  1       Feature Name    Defined
  2       Constraints     Defined
  2.1     Constraint #1   Defined
  2.1.1   Constr Type     Y Axis
  3       Flip Datum Dir  Defined
  4       Fit             Defined
  4.1     Fit Type        Default

NAME = TOP

   FEATURE IS IN LAYER(S) :
     01___PRT_ALL_DTM_PLN - OPERATION = SHOWN
     01___PRT_DEF_DTM_PLN - OPERATION = SHOWN

END ADD
```

Figure 18: Pro/Program lines for the second feature.

Pattern

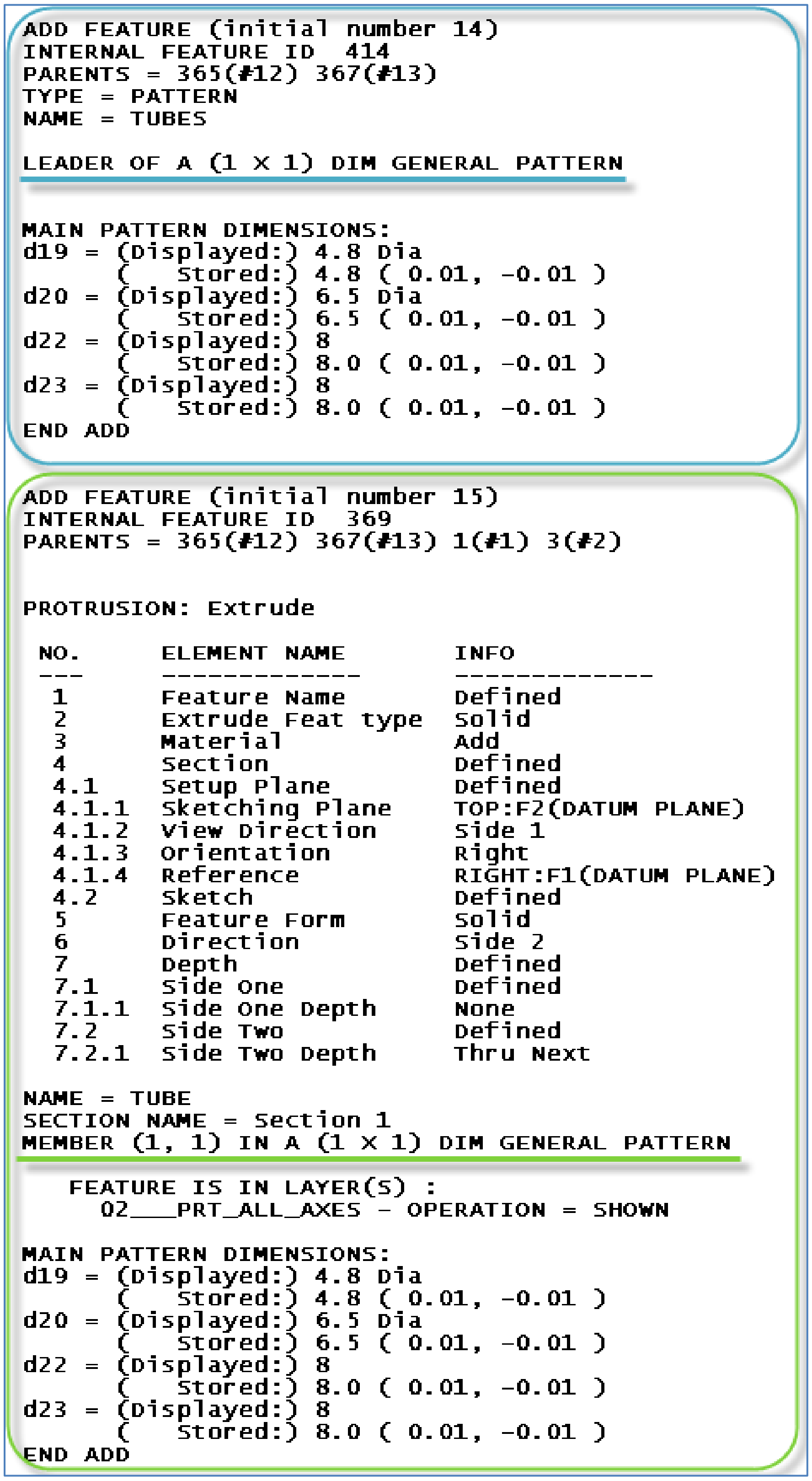
The Pro/Program block for the pattern is somewhat different. The pattern itself is a separate feature and has its own block, but all features that the pattern has (instances) are listed after the pattern block. The Pro/Program to add Tubes pattern is shown in Figure 19. The pattern can be found at the end of the Pro/Program file.

In Figure 19, the blue rectangular area is the pattern feature (TYPE = PATTERN), named TUBES and it is the leader of the pattern (LEADER OF A (1 X 1) DIM GENERAL PATTERN).

The (1 X 1) means that this pattern has only one instance; the structure is ([first pattern direction] X [second pattern direction]), f. ex. if you look at the patter of the taps (NAME = TAPS), there is showed (2 X 2). The patter feature starts with ADD and ends with END ADD.

After the pattern feature, all instances of the pattern are listed. This is showed with green rectangular area in Figure 19. Notice the line (MEMBER (1, 1) IN A (1 X 1) DIM GENERAL PATTERN) in the middle of the area. That line means, that this feature is a member of the pattern (i.e. has been patterned) and the member number is (1, 1). This pattern has only one feature, but f. ex. the TAPS pattern has four members (1,1;2,1;1,2;2,2). The last member of the pattern is the one which has the same values as the leader of the pattern. In TUBES case member (1, 1) is the last member of the pattern (1 X 1) and in the TAPS case (2, 2) is the last member in the pattern (2 X 2).

It is very important to find the leader of the pattern and the last member of the pattern, because all lines that start from the ADD of the pattern and ends whit the END ADD of the last member of the pattern are actually the whole pattern feature in the Model Tree.



```
ADD FEATURE (initial number 14)
INTERNAL FEATURE ID  414
PARENTS = 365(#12) 367(#13)
TYPE = PATTERN
NAME = TUBES

LEADER OF A (1 X 1) DIM GENERAL PATTERN


MAIN PATTERN DIMENSIONS:
d19 = (Displayed:) 4.8 Dia
      (    Stored:) 4.8 ( 0.01, -0.01 )
d20 = (Displayed:) 6.5 Dia
      (    Stored:) 6.5 ( 0.01, -0.01 )
d22 = (Displayed:) 8
      (    Stored:) 8.0 ( 0.01, -0.01 )
d23 = (Displayed:) 8
      (    Stored:) 8.0 ( 0.01, -0.01 )
END ADD
```

```
ADD FEATURE (initial number 15)
INTERNAL FEATURE ID  369
PARENTS = 365(#12) 367(#13) 1(#1) 3(#2)


PROTRUSION: Extrude

 NO.      ELEMENT NAME         INFO
 ---      -------------        -------------
  1       Feature Name         Defined
  2       Extrude Feat type    Solid
  3       Material             Add
  4       Section              Defined
  4.1     Setup Plane          Defined
  4.1.1   Sketching Plane      TOP:F2(DATUM PLANE)
  4.1.2   View Direction       Side 1
  4.1.3   Orientation          Right
  4.1.4   Reference            RIGHT:F1(DATUM PLANE)
  4.2     Sketch               Defined
  5       Feature Form         Solid
  6       Direction            Side 2
  7       Depth                Defined
  7.1     Side One             Defined
  7.1.1   Side One Depth       None
  7.2     Side Two             Defined
  7.2.1   Side Two Depth       Thru Next

NAME = TUBE
SECTION NAME = Section 1
MEMBER (1, 1) IN A (1 X 1) DIM GENERAL PATTERN

   FEATURE IS IN LAYER(S) :
     02___PRT_ALL_AXES - OPERATION = SHOWN

MAIN PATTERN DIMENSIONS:
d19 = (Displayed:) 4.8 Dia
      (    Stored:) 4.8 ( 0.01, -0.01 )
d20 = (Displayed:) 6.5 Dia
      (    Stored:) 6.5 ( 0.01, -0.01 )
d22 = (Displayed:) 8
      (    Stored:) 8.0 ( 0.01, -0.01 )
d23 = (Displayed:) 8
      (    Stored:) 8.0 ( 0.01, -0.01 )
END ADD
```

Figure 19: Pro/Program lines to add a pattern of tubes.

Next we modify the Pro/Program in that way that it allows us to suppress the pattern and its instances using the *tubes* parameter. Scroll to the end of the file and start scrolling top until you find the correct lines (initial number 19 if made according to this document). Add the **boldface** lines of the following to the appropriate positions to suppress the pattern and its instance(s):

```
if (tubes == yes)
ADD FEATURE (initial number 18)
 INTERNAL FEATURE ID  480
 PARENTS = 432(#12) 434(#13)
 TYPE = PATTERN
 NAME = TUBES

 LEADER OF A (1 X 1) DIM GENERAL PATTERN

…
[other text here]
…
NAME = TUBE
 SECTION NAME = Section 1
 MEMBER (1, 1) IN A (1 X 1) DIM GENERAL PATTERN

    FEATURE IS IN LAYER(S) :
      02___PRT_ALL_AXES - OPERATION = SHOWN

 MAIN PATTERN DIMENSIONS:
d19 = (Displayed:) 6.5 General_Dims Dia
       (   Stored:) 6.5 ( 0.2, -0.2 )
 d20 = (Displayed:) 4.8 General_Dims Dia
       (   Stored:) 4.8 ( 0.1, -0.1 )
 d22 = (Displayed:) 8 General_Dims
       (   Stored:) 8.0 ( 0.2, -0.2 )
 d23 = (Displayed:) 8 General_Dims
       (   Stored:) 8.0 ( 0.2, -0.2 )
END ADD
end if
```

Having only the lines concerning the pattern leader in the block is not sufficient; all pattern members have to be included as well. Therefore, the initial number of instances is best to be 1 x 1 if possible! Unfortunately, "end if" is written differently in Relations and Pro/Program, as shown above. The if-conditional state can also be written as "if (tubes)", but it is clearer to add "== yes" to the condition.

Save the file and exit the text editor. If the program has no errors, you are prompted for if you want to incorporate the changes into the model. Answer **Yes** and click **Done/Return**. If problems occur, click *Edit* and try to fix them. Now it should be possible to give any positive integer values to *size_h* and *size_v*, try! If you set the value of 1 to *size_h* and/or *size_*, you would notice that the TUBES are suppressed. Time to save your model?

Reading User Input

Modifying Pro/Program

Next we create inputs for the program, allowing us to easily modify parameters with every regeneration. Open **Program** (**Edit Design**) and write *these two lines* between INPUT – END INPUT.

```
...
INPUT
size_h number
"Give the length of the brick:"
END INPUT
...
```

The format of Input-field is:

[parameter's name] [parameter's type]
"[input message for the user]"

There are three input types for reading parameters:

- Number, for Integer and Real Number

- Yes_no, for boolean operator (yes/no)
- String, for text (String)

If you want more inputs, just write them between INPUT-field with the same format. Save the text in the text editor and close it. Answer **Yes** to the *Confirmation* window and click **Current Vals** from the *Menu Manager*. This regenerates our model using current dimension and parameter values. **Warning**!: do this every time after editing Input field and/or other fields, otherwise there is risk that the Pro/Program of the model is different than the Pro/Program of the text file. Also, if you just open Pro/Program with *Edit Design* and close it without doing any modifications, select *Yes* to incorporate the changes to the model, otherwise there will be two different Pro/Programs. Click **Done/Return** to close the *Menu Manager*.

Regenerating the model

To see what the lines in the Input field do, press CTRL+G to regenerate the model. The Menu Manager appears to the upper right corner of your screen. There are three options how to regenerate your model: *Current Vals* for regeneration without changing parameter values, *Enter* to give new parameter values and *Read File* to import a file where parameter values are defined. Click **Enter**, select the parameters you want to change and click **Done Sel** (Figure 20). The program gives you a text line where to give the new value for selected parameters using question line from the Input field. Give some value and see how the model regenerates.

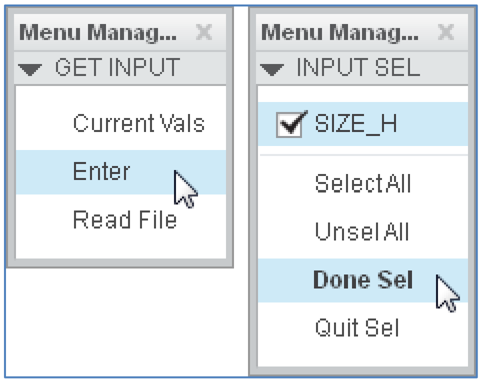


Figure 20: First selecting *Enter*, then checking *SIZE_H* and selecting *Done Sel* to regenerate the model.

Next Things to Do Without Guidance

To finalize the model, do the following tasks:

1. Create inputs for *size_v* and *thin*.
2. Create a pattern of taps (Figure 6) which are used instead of tubes if the width or length of the brick is 1. Then use the techniques you have learned to manage their number and suppress them when necessary (Figure 21). To enable value 1 for both *size_h* and *size_h*, individual patterns are needed for both directions (to be honest, not needed, but it is much easier to do this way…).

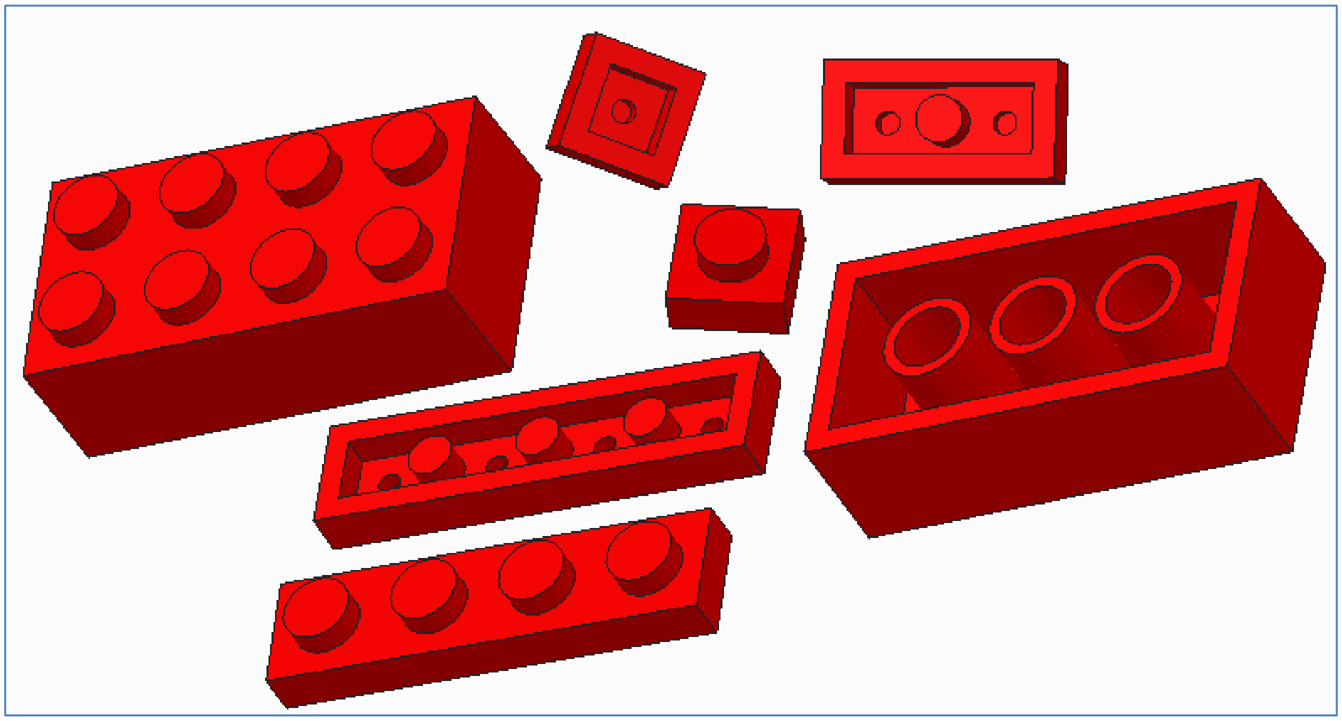


Figure 21: Different kinds of Lego bricks.

Returning

To return your model to automatic assessment system, ensure that you have three correct parameters (SIZE_V, SIZE_H and THIN).