COMP6209 Assignment Instructions

Module	COMP6209: *Automated Code Generation*			**Lecturer**	*Julian Rathke*
Assignment	Coursework 2: Template Metaprogramming			**Weight**	*20%*
Deadline	4pm 10*/5/2018*	**Feedback**	1*/6/2018*	**Effort**	*30 hours*

Instructions

The goal of the coursework is to investigate the use C++ templates to perform simple program generation tasks.

Part 1: Expression Templates

Write C++ templates which allow you to use types to represent integer arithmetic expressions over the four basic operations (addition, subtraction, multiplication, integer division), a single variable, and integer literals, for instance

x + (x-2) * (x-3)

For the purposes of Parts 1 and 2 we refer to the single variable as x.

The types representing the expressions should contain a function named 'eval' that provides code for evaluating the expression. This function should accept an integer parameter that will be used as the value for the single variable x.

Write a main() method that builds an expression as a type and evaluate it at some integer input value. Print the results to standard out.

Part 2: Bounds

Extend your expression templates from Part 1 to allow each Expression type to carry with it a lower and upper bound for the possible values the expression could evaluate to. Of course, this depends on the range of values for the single variable x.

We will assume that the bounds on the input are statically known, and therefore you should parameterise the template type for representing the variable expressions on a type BOUNDS. The type BOUNDS itself should be a template parameterised on two integer values: the lower and upper values of the range for x.

Modify the eval function to throw an exception in case the input for x is not in the specified input range.

Now modify your example in main() to build the expression based on a known range for the input values for x.

Print to standard out the calculated bounds for the output from the expression before printing the result of the evaluation.

For example, suppose that the bounds on variable x 0 <= x <= 5. Let us write this concisely as x_{0,5}. Consider a sample expression with a variable that carries these bounds:

(x_{0,5} + 3) * (x_{0,5} + 5)

Then we can calculate the bounds on the whole expression above as follows:

3 <= x_{0,5} + 3 <= 8 and 5 <= x_{0,5} + 5 <= 10

Consequently we have that

15 <= (x_{0,5} + 3) * (x_{0,5} + 5) <= 80

That is, the bounds for the above expression would be (15,80) if the bounds on x were (0,5).

For the purposes of bounds calculations, you may assume that each occurrence of x within the expression may take any input value within its input bounds. With that assumption the bounds are not as tight as they could be but will still be bounds. For example, the bounds for x * x with a bound of [-2,2] on each x will be [-4,4] (e.g. where x is -2 at the first occurrence and x is 2 for the second to get the lower bound).

Part 3: Multiple Variables

Modify your expression templates from Part 2 to allow for multiple variables in the expression. Each occurrence of a variable expression can be considered as the unique variable named x, y, z, ... For example

x + (x-2) * (x-3)

would be considered as

x + (y -2) * (z - 3).

Modify the eval function to accept an array of integer values and evaluate each variable according to the corresponding position in the input array. For instance, in the above example variable z should evaluate to the 3rd value of the input array.

Note that each variable expression should be parameterised on a BOUNDS type so that there may be different bounds for each variable.

Modify your example in main() to provide a number of example input arrays and store the results of evaluation in an array called outputs.

Part 4: Integer Declarations

Write a C++ template meta-program, IntDecl, which accepts a BOUNDS type and calculates the smallest primitive type which represents that value according to the following ranges:

Input value:
0 to 255 is type : char

0 to 65535 is type : unsignedint

-32768 to 32767 is type: int

any other integer value is type: long

Values which fall in to two or more ranges should be given the first type found in the list ordered above.

Use the calculated bounds of your example expression and the IntDecl template to declare the type for the output array in main() from Part 3.

Submission

Please submit well-documented C++ source code for each Part. That is, submit a zip archive with subdirectories called Part 1, Part 2, and Part 3. In each subdirectory have the relevant versions of the C++ templates, the example program requested and any other test programs you wish to show.

Please submit using the Handin system (http://handin.ecs.soton.ac.uk) by 4pm on the due date.

Relevant Learning Outcomes

1. *C++ Template Meta-programming*

Marking Scheme

Criterion	Description	Outcomes	Total
Part 1	*C++ template code for arithmetic expressions plus example*	*1*	6 *marks*
Part 2	*C++ template code for arithmetic expressions with Bounds plus example*	*1*	*8 marks*
Part 3	*C++ template code for arithmetic expressions with Bounds and Multi-Variables plus example*	*1*	*4 marks*
Part 4	*C++ template code for declaring Integers within Ranges*	1	2 *marks*

Late submissions will be penalised at 10% per working day. No work can be accepted after feedback has been given. Please note the University regulations regarding academic integrity.