UNIVERSITY OF Southampton
School of Electronics and Computer Science

COMP6209 Assignment Instructions

Module	COMP6209: *Automated Code Generation*			**Lecturer**	*Julian Rathke*
Assignment	Coursework 1: Template Metaprogramming			**Weight**	*20%*
Deadline	4pm 12*/3/2020*	**Feedback**	*3/4/2019*	**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 polynomial expressions using the five operations (addition, subtraction, multiplication, division and integer exponentiation), a single variable, and integer literals, for instance

x^4 + (x-2)^2 + (x*3) + 5

For the purposes of this assignment 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 a double 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 double input value. Print the results to standard out.

Part 2: Numerical Integration

Use the expression templates from Part 1 to write a C++ template that accepts polynomial expressions and generates code for numerically approximating the integral of that expression with respect to the single variable. Your template should provide a function named 'integrate' that accepts double values as the lower and upper limits of the range to be integrated over. The integrate function should also accept an integer value that is known at compile time that specifies the number of divisions of the range by which the integral is approximated. As well as this, your template should accept the interpolation function to be used to approximate each interval. A good solution will statically unroll any loops required.

Write a main() method that shows how to use this template to generate code for a short example program that numerically integrates the arithmetic expression:

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

for both the rectangle rule and trapezoidal rule over the interval [0, 10] with 200 divisions of the range. Write the results to standard out.

(see https://en.wikipedia.org/wiki/Numerical_integration#Quadrature_rules_based_on_interpolating_functions) for details on numerical integration and interpolation functions.

Part 3: Symbolic Differentiation

Write a series of C++ templates that implement a meta-function ‘derivative’ over templates that takes an expression from Part 1 and returns an expression representing the derivative of the input expression with respect to the single variable x. Recall, that derivatives of polynomials are calculated as follows:

$$(e + f)' = (e') + (f')$$

$$(e * f)' = (e * (f)') + ((e)' * f)$$

$$(e - f)' = (e)' - (f)'$$

$$(e / f)' = (f * (e)' - e * (f)') / f^2$$

$$(e^n)' = n * e^{n-1} * (e)'$$

Show how to use these templates to generate code for a short example program that calculates the value of the derivative of:

$$(x^{20}+3) / (x-3)$$

at values x = 5 and x = 10.

Part 4: Integer Declarations

Write a C++ template meta-program, IntDecl, that accepts an integer polynomial expression that **does not contain any use of the division operator** and a statically known integer value called X. IntDecl should calculate the smallest primitive type that can be used to declare a variable that is capable of storing both the value of the polynomial expression or its derivative evaluated at X. Note that you will not be able to use the evaluation function from Part 1 to statically evaluate the polynomial and its derivative. Implement a series of C++ templates called EVAL parameterised on the static integer X and the polynomial expression to achieve this instead. Use the following ranges as a guide:

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 expression $4(x^3) + 2(x^2) + 50$ and the IntDecl template at points -30,-20,-10,0,10,20, and 30 to declare different variables with the smallest primitive type.

Submission

Please submit well-documented C++ source code for each Part. That is, submit a zip archive with subdirectories called Part 1, Part 2, Part 3 and Part 4. 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*	5 *marks*
Part 2	*C++ template code for numerical integration*	*1*	6 *marks*
Part 3	*C++ template code for calculating derivatives*	*1*	5 *marks*
Part 4	*C++ template code for declaring Integers within Ranges*	1	4 *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.