Homework 1

Due: Tue, Feb 14, 2017

In this assignment, you will be creating an internal language representation for a small expression language.

1 Language

The language has the following kinds of types:

$$\begin{array}{rcl} t & ::= & \mathsf{bool} \\ & & \mathsf{int} \end{array}$$

The Boolean type bool describes the values true ($\top$) and false ($\bot$). The integer type int type describes integer values in the left-open range $[-2^{32-1}, 2^{32-1} - 1)$.

The language has the following expressions.

e	$::=$	true		$e_1 < e_2$	less than
		false		$e_1 > e_2$	greater than
		e_1 and e_2	and	$e_1 \leq e_2$	less than or equal to
		e_1 or e_2	inclusive or	$e_1 \geq e_2$	greater than or equal to
		e_1 or e_2	exclusive or	$e_1 + e_2$	addition
		not e	logical negation	$e_1 - e_2$	subtraction
		if e_1 then e_2 else e_3	conditional	$e_1 * e_2$	multiplication
		$e_1 = e_2$	equal to	e_1 div e_2	integer division
		$e_1 \neq e_2$	not equal to	e_1 rem e_2	remainder of division
		$e_1 < e_2$	less than	$-e_1$	arithmetic negation

An expression is a sequence of operands and operators that specifies a computation. The evaluation of an expression results in a value. The type of an expression determines how expressions can be combined to produce complex compuations and the kind of value it produces. The following paragprahs define the operand requirement and result types of each expression as well the values they produce.

The order in which operands of an expression is unspecified unless otherwise noted.

The expressions true and false have type bool and the values $\top$ and $\bot$, respectively.

The operands of the expressions e_1 and e_2, e_1 or e_2, and e_1 or e_2 shall have type bool. The result type of each is bool. The result of e_1 and e_2 is true if both operands are $\top$ and false otherwise. The result of e_1 or e_2 is false if both operands are $\bot$ and true otherwise. The result of e_1 or e_2 is false if both operands are equal and true otherwise.

The operand of not e shall have type bool, and the type of the expression is bool. The result of the expression is true when the operand is $\bot$ and false otherwise.

In the expression if e_1 then e_2 else e_3, the type of e_1 shall be bool, and e_2 and e_3 shall have the same type. The type of the expression is the type of e_2 and e_3. The result of expression is determined by first evaluating e_1. If that value is $\top$ then the result of the expression is the value of e_2, otherwise, it is the value of e_3. Only one of e_2 or e_3 is evaluated.

The operands of the expressions $e_1 = e_2$ and $e_1 = e_2$ shall have the same type. The result type is bool. The result of $e_1 = e_2$ is $\top$ if e_1 and e_2 have equal values and $\bot$ otherwise. The result of $e_1 \neq e_2$ is $\bot$ if e_1 and e_2 have equal values and $\neq$ otherwise. Note that for bool operands, this is equivalent to the expression e_1, e_2or.

The operands of the expressions $e_1 < e_2$, $e_1 > e_2$, $e_1 \leq e_2$, and $e_1 \geq e_2$ shall have type int. The result type is bool. The result of $e_1 < e_2$ is $\top$ if e_1 is less than e_2 and $\bot$ otherwise. The result of $e_1 > e_2$ is $\top$ if e_1 is greater than e_2 and $\bot$ otherwise. The result of $e_1 \leq e_2$ is $\top$ if e_1 is less than or equal to e_2 and $\bot$ otherwise. The result of $e_1 \geq e_2$ is $\top$ if e_1 is greater than or equal to e_2 and $\bot$ otherwise.

The operands of the expressions $e_1 + e_2$, $e_1 - e_2$, $e_1 * e_2$, and e_1 div e_2 shall have type int. The result type is int. The result of $e_1 + e_2$ is the sum of the operands. If the sum is greater than the maximum value of int, the result is undefined. The result of $e_1 + e_2$ is the difference resulting from the subtraction of the e_2 from e_1. If the difference is less than the minimum value of int, the result is undefined. The result of $e_1 * e_2$ is the product of the operands. If the product is greater than the maximum value of int, the result is undefined. The results of e_1 div e_2 and e_1 rem e_2 are the quotient and remainder of dividing e_1 by e_2, respectively. In either case, if e_2 is 0, the result is undefined. If e_2 is the minimum value of int, the result is undefined. For division, the fractional part of the value is discarded (the value is truncated towards zero). If the expression a div b is defined, $(a \text{ div } b) * b + a \text{ rem } b$ is equal to a.

The operand of $-e_1$ shall have type int, and the type of the expression is int. The result of the expression is the arithmetic inverse of the value of e_1. This is equivalent to the value of $0 - e_1$.

2 Requirements

Implement a C++ library (collection of classes and functions) that defines this language. Prefer to represent the expressions as a class hierarchy. The evaluation function can be implemented as a simple virtual function.

Write a test program that create and evaluate a number of expressions. There should be at least one test for each expression.

Store your work in an online repository. I recommend GitHub, GitLab, or BitBucket.

Write a short paper to serve as an overview and guide for your implementation. Describe the components of your project in a way that would help a newcomer understand the project's organization. Be sure to include a link to your online source code in your submission.

Submission: Submit your printed homework on the due date. Send me an email with a link to your online source code.

Above and beyond: Make sure that the classes in your hierarchy are well constructed (constructors, access restrictions, appropriate documentation). Note that the semantics of the language could be written

Consider adding two extra expressions:

$$\begin{array}{lcl} e & ::= & \cdots \\ & & e_1 \ \textsf{and then} \ e_2 \\ & & e_1 \ \textsf{or else} \ e_2 \end{array}$$

The expression e_1 and then e_2 is equivalent to if e_1 then e_2 else false. The expression e_1 or else e_2 is equivalent to if true then e_2 else e_2.