Universidade de Brasília

Gianlucas Dos Santos Lopes - 180041991

Ricardo de Castro Giometti Santos - 180027263

ExpressionPL

This project follows the article "Evaluating Support for Features in Advanced Modularization Technologies" by Lopez-Herrejon, Roberto & Batory, Don & Cook, William. (2005).

In this code we implement the Expression Problem in c++ using template metaprogramming together with mixins.

Requirements

This code has been tested with the following specifications:

- g++ or clang C++17 compiler
- Google Test library
- GNU Make

Files

The project is divided in two folders. ***src*** contains the implementation of the six modules proposed in the article that can be used to compose different programs as listed. 2 simple implementations of this programs can be found in the folder src/programs.

In the folder src/programs/example, we have an example of how flexible this implementation can be. The objective is to split the lp module(lp.cpp) into two different modules, one implementing the Interface Exp and Lit datatype with no functionality (lit.cpp) and other module that can be used to add the Print functionality to the first module (print.cpp). example_main.cpp uses this two new classes, and shows that we can use both of them together with the old module le.cpp, without needing to change code.

Folder ***tests*** contains unit tests using Google Test framework (needs separate installation), which test functionalities from all src modules (alone and combined).

Compile and Run

To compile everything, from the ExpressionPL folder, do:

```
$ make
```

To run tests do:

```
$ ./tests/test
```

To run the example do:

```
$ ./src/programs/example/example_main
```

Other programs (listed in the article)

```
$ ./src/programs/program1
$ ./src/programs/program6
```

Object creation notes

In object creation, all Interfaces should be inherited before adding the classes and class deltas: Can't do

```
LitEval<ExpEval<Lit<Exp>>>
```

Also, class deltas should never be inherited before the class defition : Can't do

```
LitEval<ExpEval>
```

or

```
Lit<LitEval<ExpEval<Exp>>>
```