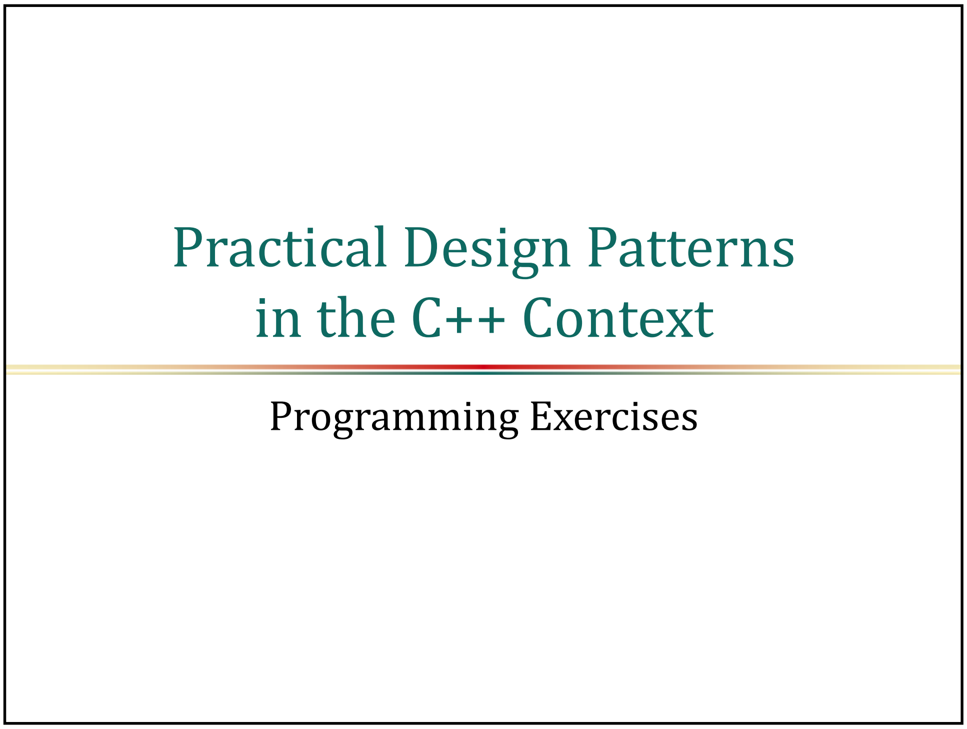
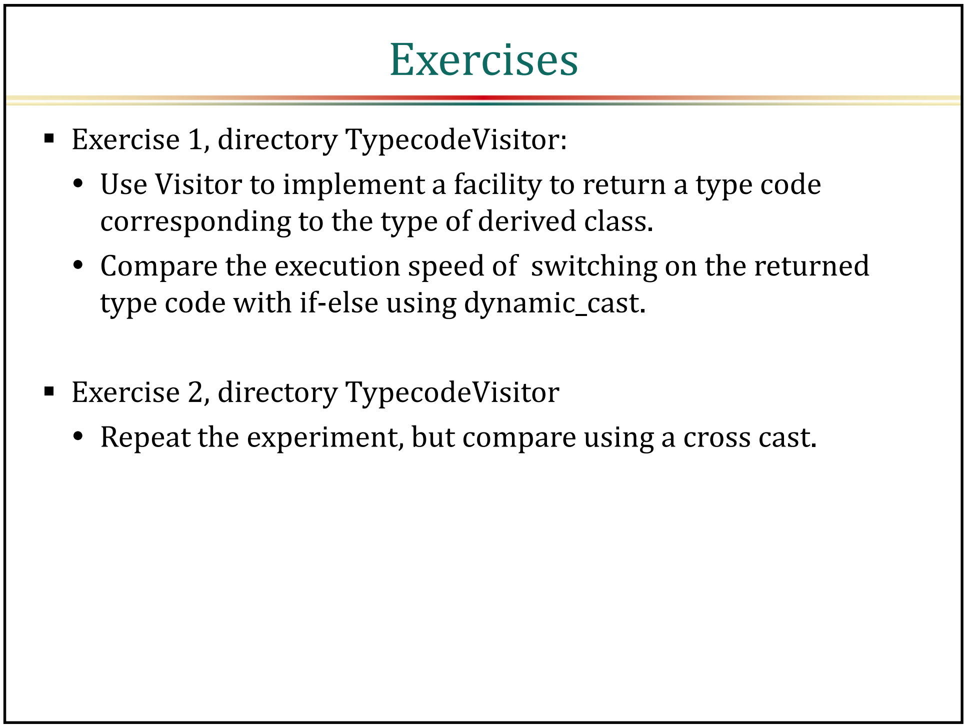


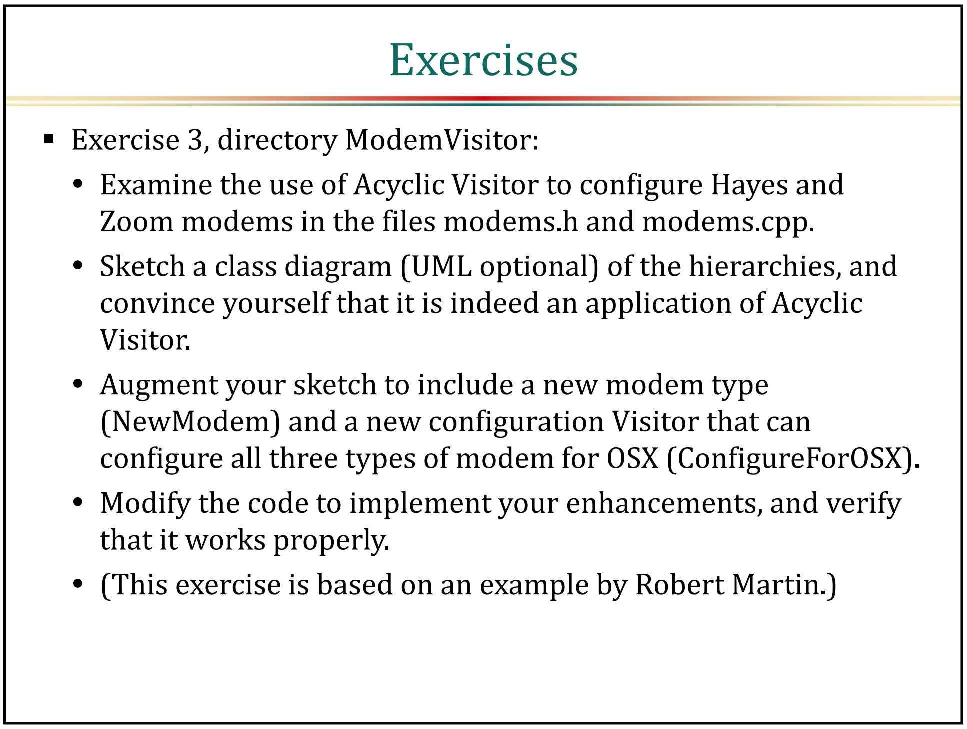
Practical Design Patterns in the C++ Context

Programming Exercises



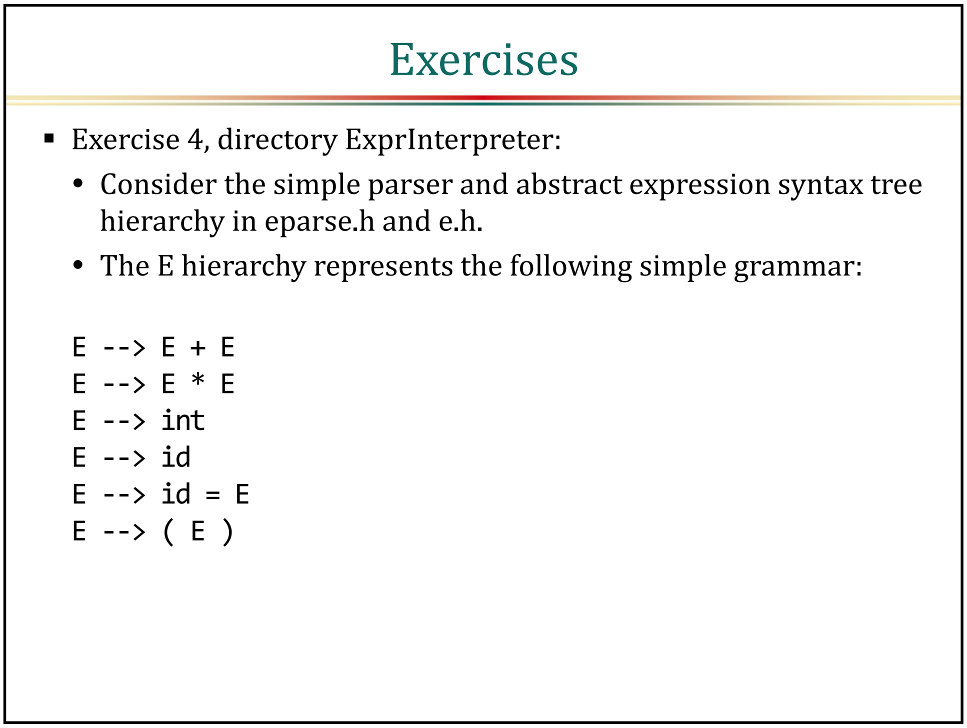
Exercises

- Exercise 1, directory TypecodeVisitor:
  - Use Visitor to implement a facility to return a type code corresponding to the type of derived class.
  - Compare the execution speed of switching on the returned type code with if-else using dynamic_cast.
- Exercise 2, directory TypecodeVisitor
  - Repeat the experiment, but compare using a cross cast.



Exercises

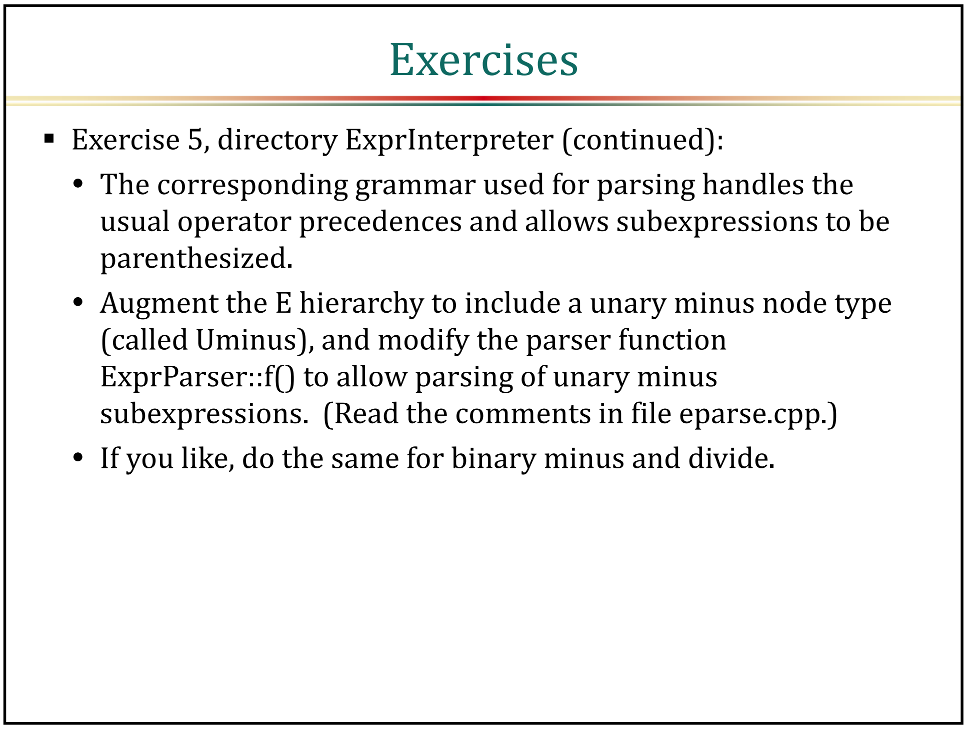
- Exercise 3, directory ModemVisitor:
  - Examine the use of Acyclic Visitor to configure Hayes and Zoom modems in the files modems.h and modems.cpp.
  - Sketch a class diagram (UML optional) of the hierarchies, and convince yourself that it is indeed an application of Acyclic Visitor.
  - Augment your sketch to include a new modem type (NewModem) and a new configuration Visitor that can configure all three types of modem for OSX (ConfigureForOSX).
  - Modify the code to implement your enhancements, and verify that it works properly.
  - (This exercise is based on an example by Robert Martin.)



Exercises

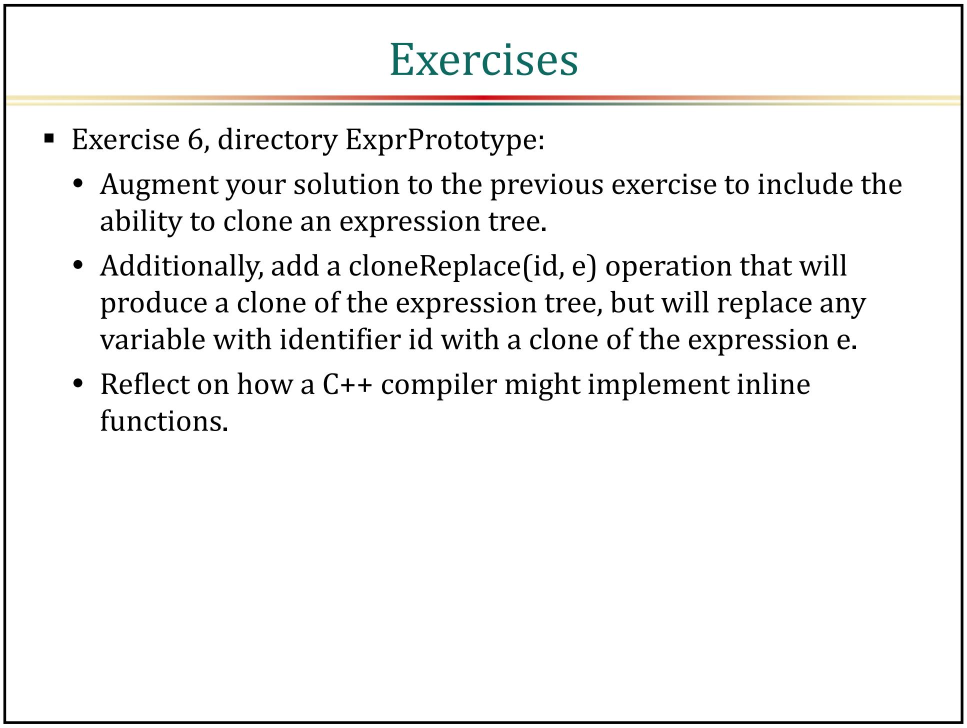
- Exercise 4, directory ExprInterpreter:
  - Consider the simple parser and abstract expression syntax tree hierarchy in eparse.h and e.h.
  - The E hierarchy represents the following simple grammar:

```
E --> E + E
E --> E * E
E --> int
E --> id
E --> id = E
E --> ( E )
```



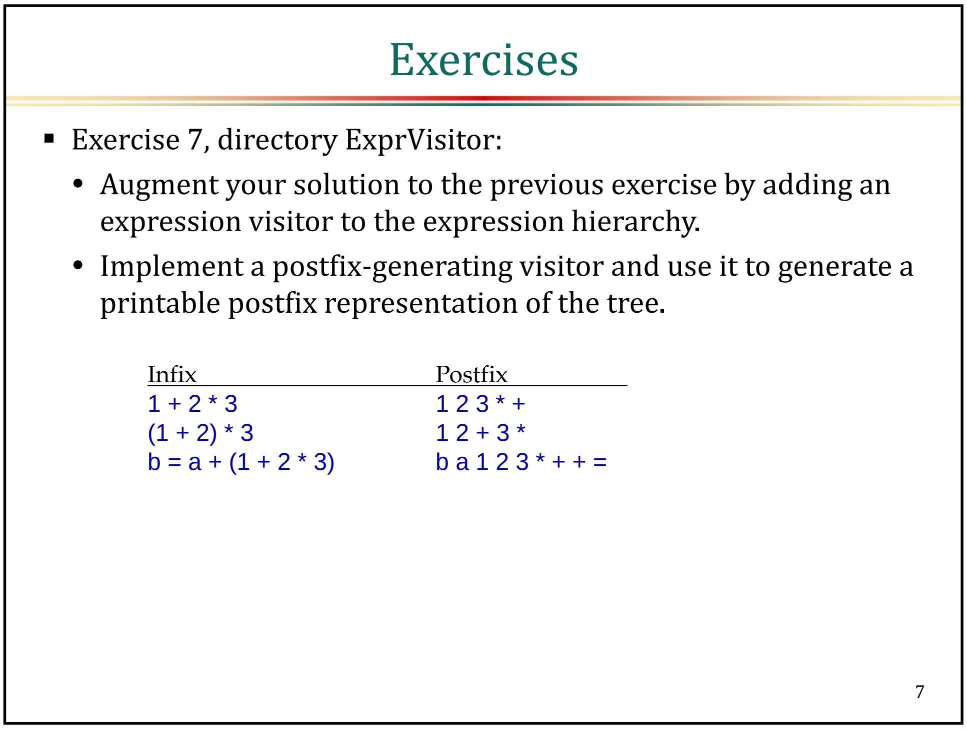
Exercises

- Exercise 5, directory ExprInterpreter (continued):
  - The corresponding grammar used for parsing handles the usual operator precedences and allows subexpressions to be parenthesized.
  - Augment the E hierarchy to include a unary minus node type (called Uminus), and modify the parser function ExprParser::f() to allow parsing of unary minus subexpressions. (Read the comments in file eparse.cpp.)
  - If you like, do the same for binary minus and divide.



Exercises

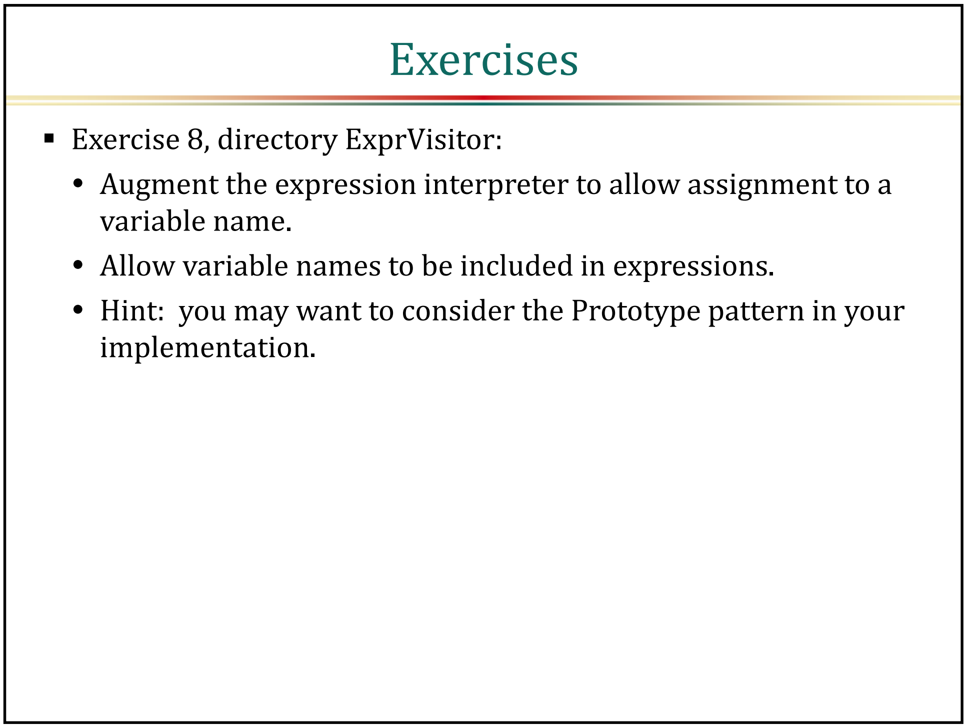
- Exercise 6, directory ExprPrototype:
  - Augment your solution to the previous exercise to include the ability to clone an expression tree.
  - Additionally, add a cloneReplace(id, e) operation that will produce a clone of the expression tree, but will replace any variable with identifier id with a clone of the expression e.
  - Reflect on how a C++ compiler might implement inline functions.



Exercises

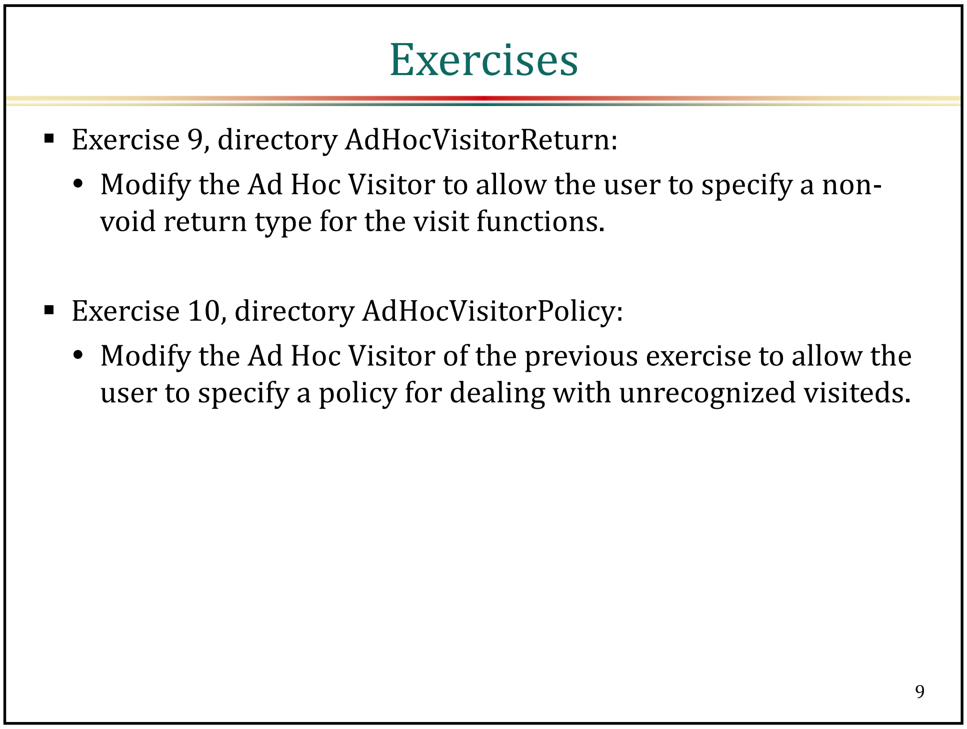
- Exercise 7, directory ExprVisitor:
  - Augment your solution to the previous exercise by adding an expression visitor to the expression hierarchy.
  - Implement a postfix-generating visitor and use it to generate a printable postfix representation of the tree.

Infix	Postfix
1 + 2 * 3	1 2 3 * +
(1 + 2) * 3	1 2 + 3 *
b = a + (1 + 2 * 3)	b a 1 2 3 * + + =



Exercises

- Exercise 8, directory ExprVisitor:
  - Augment the expression interpreter to allow assignment to a variable name.
  - Allow variable names to be included in expressions.
  - Hint: you may want to consider the Prototype pattern in your implementation.



Exercises

- Exercise 9, directory AdHocVisitorReturn:
  - Modify the Ad Hoc Visitor to allow the user to specify a non-void return type for the visit functions.
- Exercise 10, directory AdHocVisitorPolicy:
  - Modify the Ad Hoc Visitor of the previous exercise to allow the user to specify a policy for dealing with unrecognized visiteds.

9