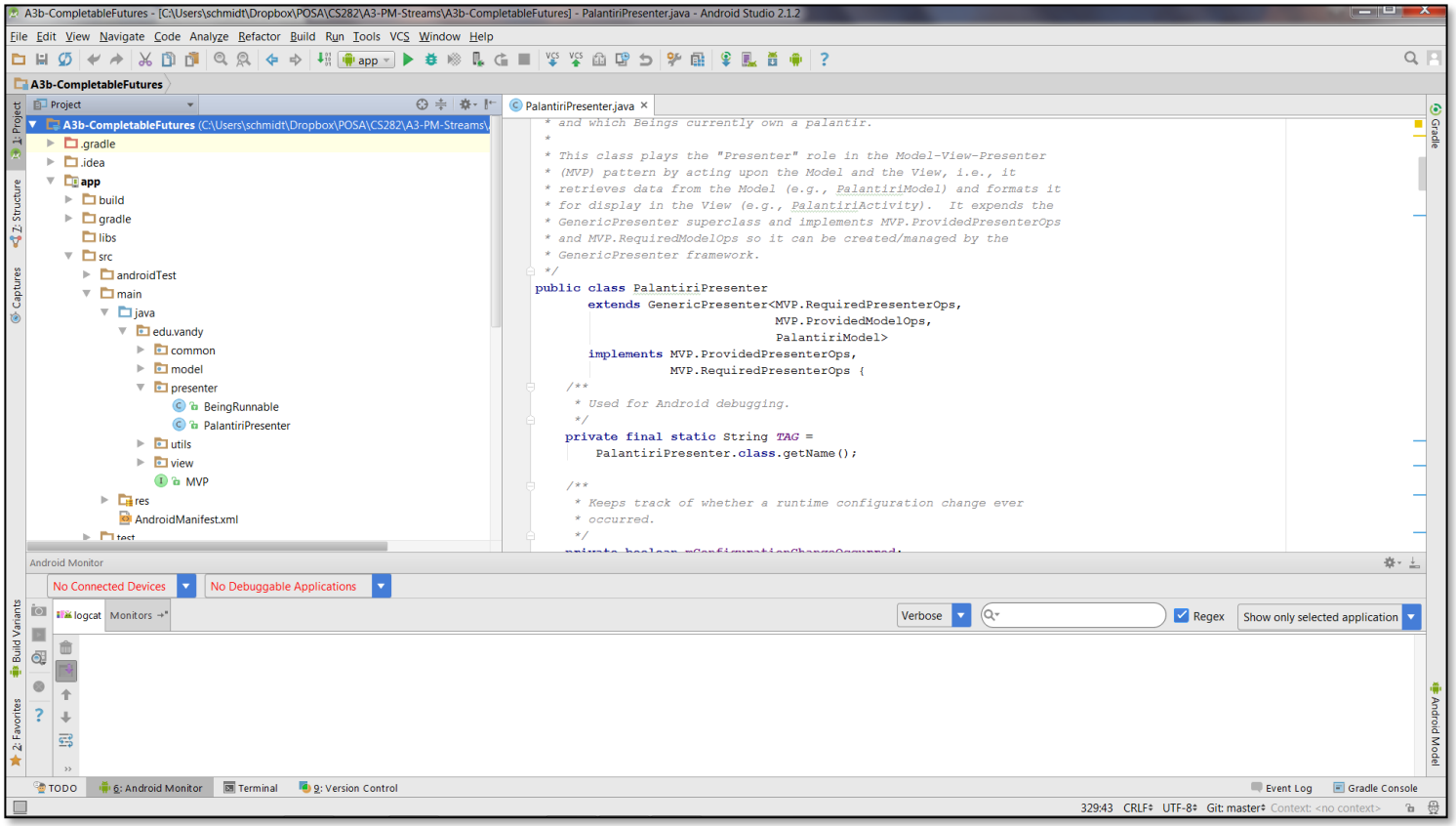
The Iterator Pattern

Implementation in C++

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Iterator* pattern can be applied to access all nodes in an expression tree flexibly & extensibly.
- Understand the structure & functionality of the *Iterator* pattern.
- Know how to implement the *Iterator* pattern in C++.



Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

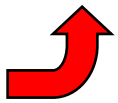
```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  stack <Expression_Tree> stack_;

  Pre_Order_ET_Iter_Impl(const Expression_Tree &tree)
    : ET_Iter_Impl(tree),
      stack_() {
    if(!tree_.is_null()) stack_.push(tree);
  }

  Expression_Tree operator *() { return stack_.top(); }
  ...
```

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  stack <Expression_Tree> stack_;
```

Pre_Order_ET_Iter_Impl inherits from ET_Iter_Impl



```
  Pre_Order_ET_Iter_Impl(const Expression_Tree &tree)
    : ET_Iter_Impl(tree),
      stack_() {
    if(!tree_.is_null()) stack_.push(tree);
  }


  Expression_Tree operator *() { return stack_.top(); }
  ...
```

See docs.oracle.com/javase/8/docs/api/java/util/Iterator.html

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  stack <Expression_Tree> stack_;
```

The stack keeps track of nodes that remain to be processed

```
  Pre_Order_ET_Iter_Impl(const Expression_Tree &tree)
    : ET_Iter_Impl(tree),
      stack_() {
    if(!tree_.is_null()) stack_.push(tree);
  }


  Expression_Tree operator *() { return stack_.top(); }
  ...
```

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  stack <Expression_Tree> stack_;
```

```
  Pre_Order_ET_Iter_Impl(const Expression_Tree &tree)
    : ET_Iter_Impl(tree),
      stack_() {
    if(!tree_.is_null()) stack_.push(tree);
  }
```

Constructor initializes the iterator

```
  Expression_Tree operator *() { return stack_.top(); }
  ...
```

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  stack <Expression_Tree> stack_;
```

```
  Pre_Order_ET_Iter_Impl(const Expression_Tree &tree)
    : ET_Iter_Impl(tree),
      stack_() {
    if(!tree_.is_null()) stack_.push(tree);
  }
```

Dereference the current iterator item

```
  Expression_Tree operator *() { return stack_.top(); }
  ...
```

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  ...
  bool
  Pre_Order_ET_Iter_Impl::operator==(const ET_Iter_Impl &rhs) {
    if (auto rhs_cast = dynamic_cast <decltype(this)>(&rhs)) {
      auto &t1 = lhs->tree_, &t2 = rhs_cast->tree_;
      auto &s1 = lhs->stack_, &s2 = rhs_cast->stack_;

      if (t1.get_root () == t2.get_root ()
          && s1.size () == s2.size ()) {
        if (s1.empty () && s2.empty ())
          return true;
        if (s1.top ().get_root () == s2.top ().get_root ())
          return true;
      }
    }
    return false; ...
```

Comparing iterators for equality

Comparing C++ STL iterators for equality can be complex!

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  ...
```

Advance iterator by one

```
  void
  Pre_Order_ET_Iter_Impl::operator++ () {
    if (!stack_.empty ()) {
      Expression_Tree current = stack_.top ();
      stack_.pop();

      if (!current.right ().is_null ())
        stack_.push (current.right ());
      if (!current.left ().is_null ())
        stack_.push (current.left ());

  }
  ...
```

The use of a stack simulates recursion, one item at a time.

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

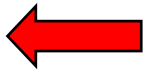
```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  ...

  void
  Pre_Order_ET_Iter_Impl::operator++ () {
    if (!stack_.empty ()) {
      Expression_Tree current = stack_.top ();
      stack_.pop();

      if (!current.right ().is_null ())
        stack_.push (current.right ());
      if (!current.left ().is_null ())
        stack_.push (current.left ());

}
...
```

Get the next item

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  ...

  void
  Pre_Order_ET_Iter_Impl::operator++ () {
    if (!stack_.empty ()) {
      Expression_Tree current = stack_.top ();
      stack_.pop();

      if (!current.right ().is_null ())
        stack_.push (current.right ());
      if (!current.left ().is_null ())
        stack_.push (current.left ());

  }
  ...
```

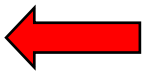


Remove current item from stack

Iterator example in C++

- A **`stack`** implements a non-recursive "pre-order" algorithm for tree traversal.

```
class Pre_Order_ET_Iter_Impl : public ET_Iter_Impl {
  ...

  void
  Pre_Order_ET_Iter_Impl::operator++ () {
    if (!stack_.empty ()) {
      Expression_Tree current = stack_.top ();
      stack_.pop();

      if (!current.right ().is_null ())
        stack_.push (current.right ());
      if (!current.left ().is_null ())
        stack_.push (current.left ());

  }
  ...
```



Update the stack

Iterator example in C++

- Implement the **`begin()`** factory method in the **`Expression_Tree`** class to return the designated iterator .

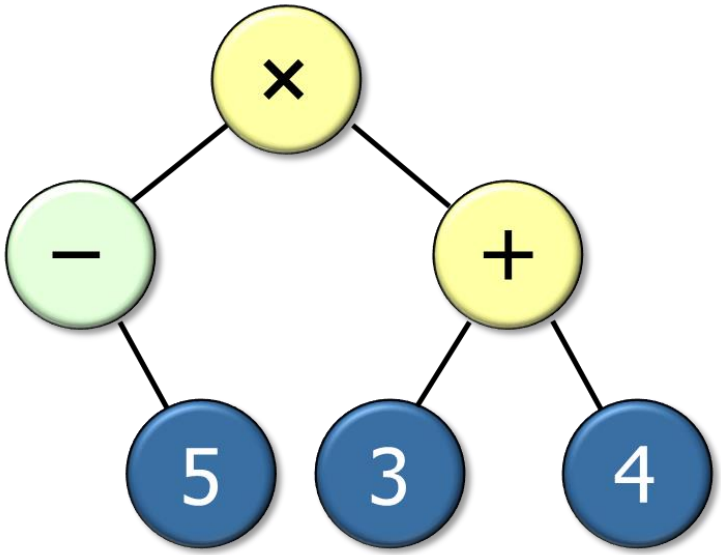
```
class Expression_Tree {
  ...
  iterator begin(const string &order) {
    return Expression_Tree::iterator(tree_iterator_factory.
      make_iterator (*this, traversal_order, false));
  }
}
```

This is an application of the *Factory Method* pattern.

We'll cover other *Iterator* implementations in the *Strategy* pattern discussion.

Iterator example in C++

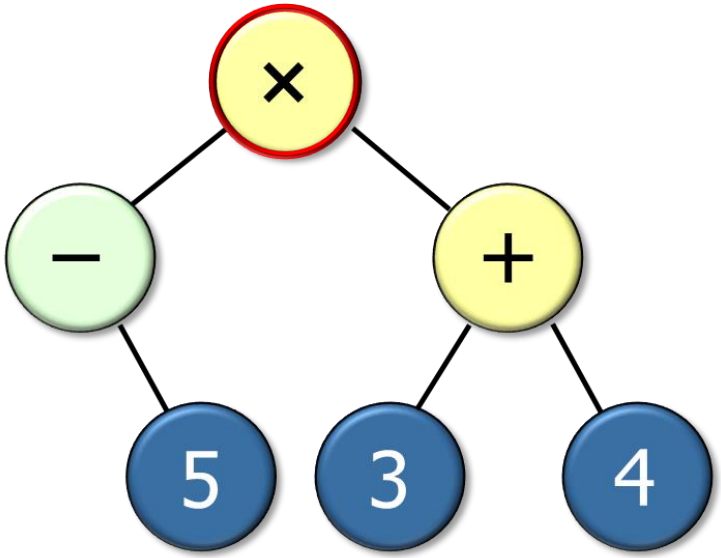
- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.




```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

Iterator example in C++

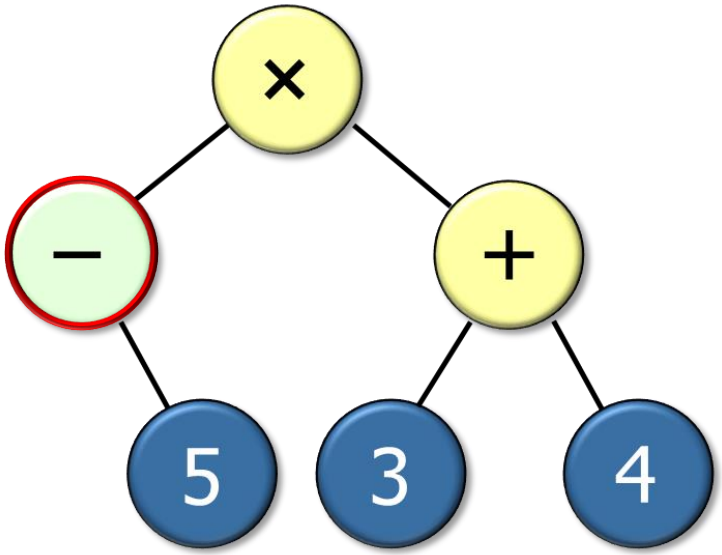
- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.



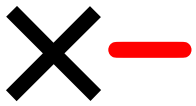

```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

Iterator example in C++

- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.

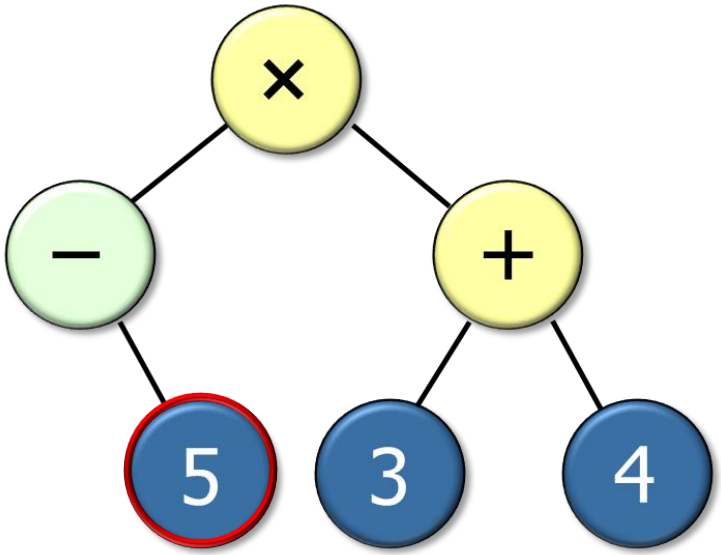



```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```



Iterator example in C++

- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.

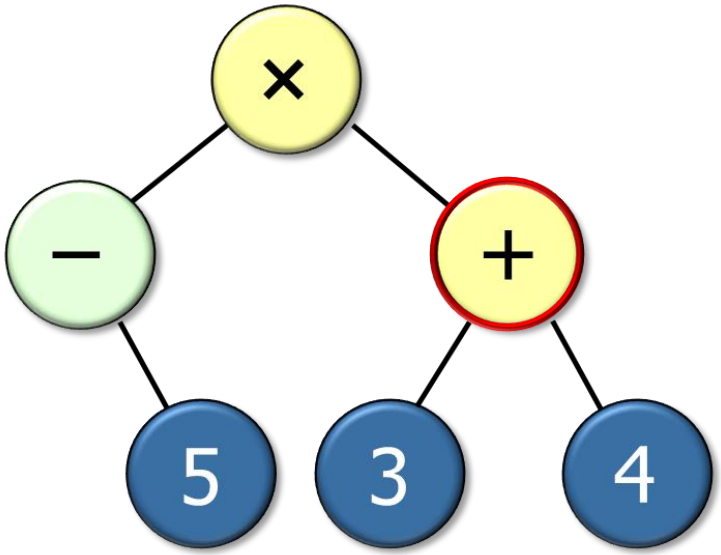



```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

×−5

Iterator example in C++

- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.

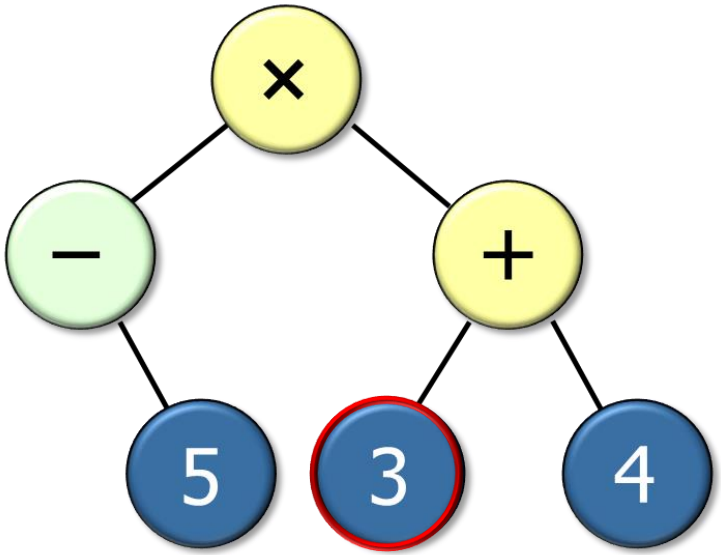



```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

× − 5 +

Iterator example in C++

- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.

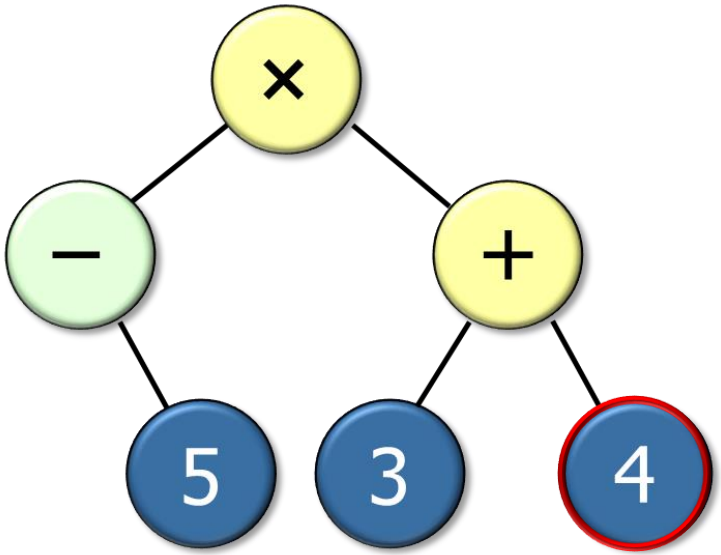



```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

×−5+3

Iterator example in C++

- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.

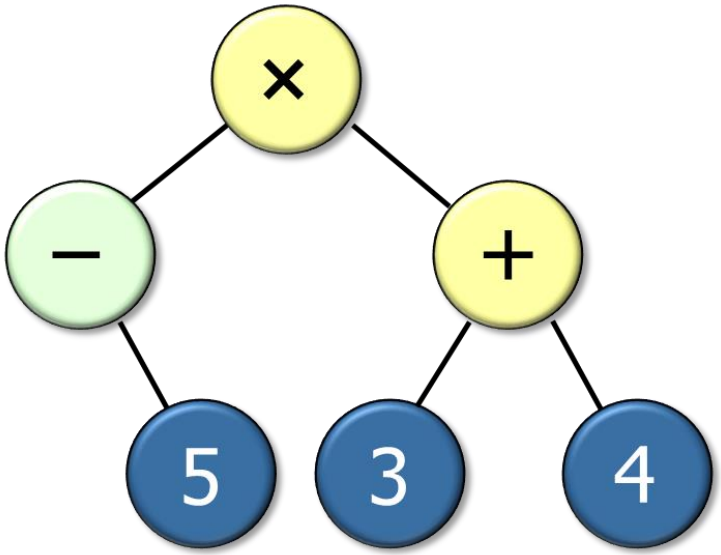



```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

×−5+34

Iterator example in C++

- Use **`Pre_Order_ET_Iter_Impl`** to print out the contents of an expression tree.




```
Expression_Tree expr_tree = ...;

cout << "Tree contents:" << endl;

for (auto iter = tree.begin(order);
     iter != tree.end(order);
     ++iter) {
   Expression_Tree node = *iter;
   if (dynamic_cast<Leaf_Node *> (node.get_root())
     cout << (int) node.item() + " ";
   else
     cout << (char) node.item() + " ";
}
```

Later we show how the *Visitor* pattern can eliminate the use of dynamic casts!