Lecture 25 Project 4

- Objectives:
  - Develop and use a Tree ADT – n-ary
  - Appy and use tree traversal algorithms
  - Manipulate trees by inserting nodes
  - Use parts of the STL

- Develop a program to instrument C++ source code to support program profiling

Building a Profiler

- Need to automatically insert counters (source code) to keep track of how many times a statement (line) has been executed

- `x = x + 1;  foo.count(__LINE__);`
- Foo is an object to keep track of how many times the line is executed
- count is a method to increment
- __LINE__ is a C++ macro to get the line number

What is Needed?

- Need to find all the statement that are to be profiled
- Expression statements – calls, assignments, operations

- Need to parse the source code to find all these statements
- Parsing the code results in an Abstract Syntax Tree (AST)
- Need to search this tree for the statements to profile
- Insert a node (source code) into the tree after each statement

Parsing C++

- The most difficult part of building a profiler is the parsing
- C++ is very difficult to parse (notoriously so)

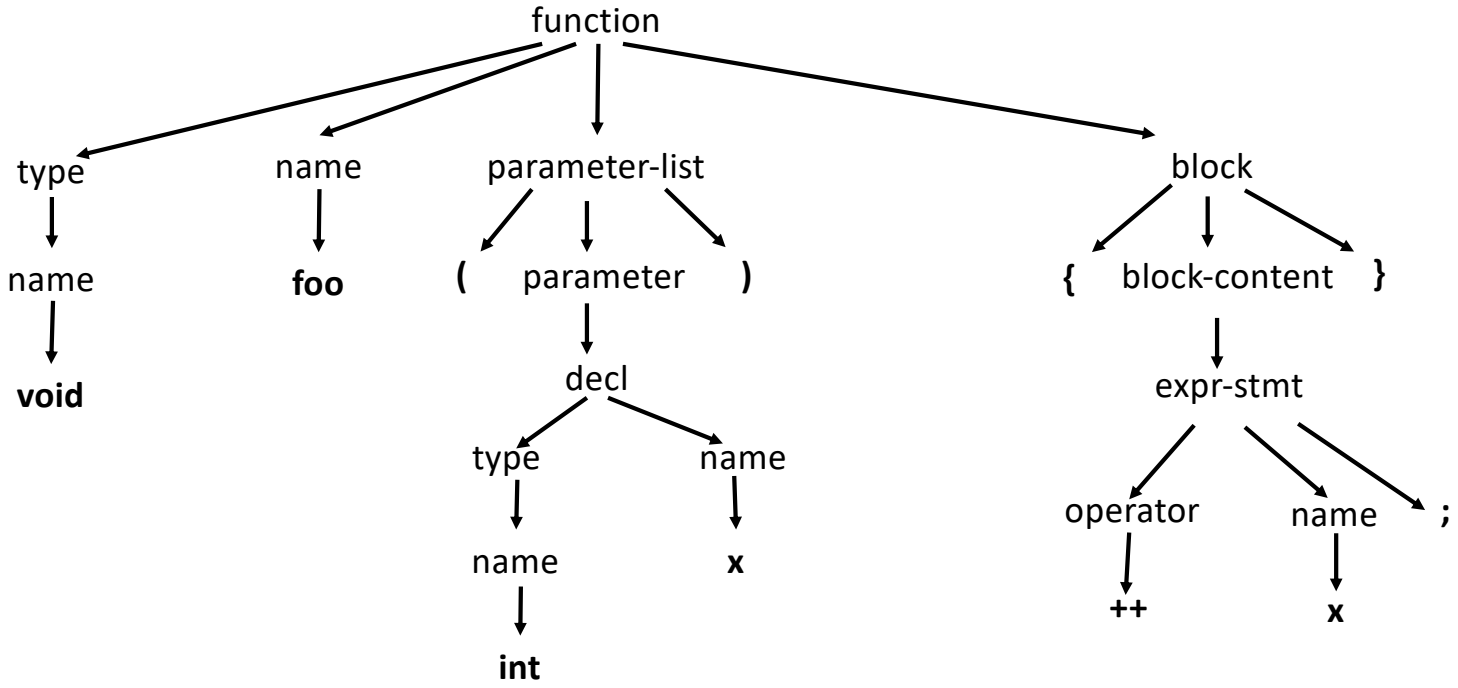
- Will use a parsing tool call srcML - generates an AST in XML
- srcML.org
- Developed by Maletic and Collard (U Akron) along with grad students
- Very robust - used in both research and industry

Abstract Syntax Tree (AST)

- Tree representation of the syntactic structure of the source code
- Based on the grammar for the programming language
- Describes the syntactic category of each language construct

```
void foo(int x) {
    ++x;
}
```

AST


function
type
name
parameter-list
block
name
foo
(
parameter
)
{
block-content
}
void
decl
expr-stmt
type
name
operator
name
;
name
x
++
x
int

srcML[1.0]

```
<function>
    <type><name>void</name></type>
    <name>foo</name>
    <parameter_list> (
        <parameter>
              <decl>
                  <type><name>int</name></type>
                  <name>x</name></decl></parameter>
         ) </parameter_list>
    <block> {<block_content>
        <expr_stmt><expr>
            <operator>++</operator><name>x</name>
        </expr>; </expr_stmt>
    </block_content>}</block>
</function>
```

Basic Requirements

- Implemented and running:
  - Read in one or more srcML documents
  - Build an internal tree representation of the AST
  - Write out the AST as source code
- Need to complete:
  - Read and understand the provided code
  - Build the copy ctor and dtor
  - Traverse tree to find specific statements
  - Instrument the code