

CS352 Compilers: Principle and Practice Spring 2005

Prof. Tony Hosking

PSO Session on Monday, January 24, 2005 *

TA: Mummoorthy Murugesan
mummoorthy@cs.purdue.edu

Office Hours: Thursday 3:00-5:00pm, PHYS 92
Friday 2:00-3:00pm, PHYS 92

* Course web-link

<http://www.cs.purdue.edu/homes/hosking/352/index.html>

*You can find the materials I cover during PSO at:

<http://www.cs.purdue.edu/homes/mmuruges/cs352/cs352.htm>

*Check the newsgroup for IMPORTANT messages: **purdue.class.cs352**

* Adapted from Bernice's Jan 18 notes

Today's topic: JavaCC Tutorial and Project 1

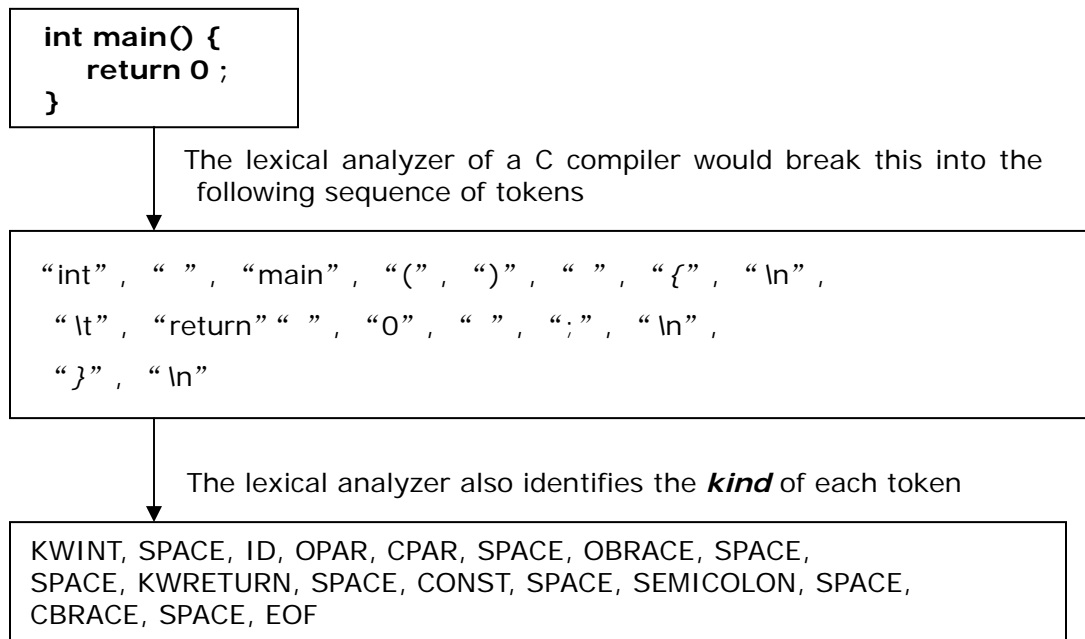
1. Outline

- Secrets for doing well in this course (during the PSO)
- PSO Structure
- Review: Lexical analyzers & Parser
- JavaCC Tutorial
- Problems to work on
- Project 1 - **Due 11:59pm Friday 4th February**
- Perfect Programming 1

2. Review: Lexical analyzers

- Lectures are available online
<http://www.cs.purdue.edu/homes/hosking/352/notes/01-intro.pdf>
<http://www.cs.purdue.edu/homes/hosking/352/notes/02-lex.pdf>
- Lexical analyzers can break a sequence of characters into subsequences called *tokens* and it also classifies the tokens.

Consider the following C program:



E.g.: Howareyoudoing -> How are you doing

3. Review: Parser

- The sequence of tokens is then passed on to the parser.
- The parser then analyses the sequence of tokens to determine the structure of the program
- Basically, it checks the program with certain grammatical rules.

4. JavaCC

JavaCC itself is not a parser or a lexical analyzer but a **generator**. This means that it outputs lexical analyzers and parser according to a specification that it reads in from a file. JavaCC produces lexical analyzers and parsers written in Java.

4.1 JavaCC Configuration – Set environment variables for JavaCC

<http://www.cs.purdue.edu/homes/hosking/352/project/project0.html>

4.2 A simple Example – Adder (Parser does nothing)

```
PARSER_BEGIN(Adder)
class Adder{
    static void main(String args[]) throws ParseException,
    TokenMgrError
    {
        Adder parser=new Adder(System.in);
        parser.Start();
    }
}
PARSER_END(Adder)
```

```
SKIP: { " " }
SKIP: { "\n" | "\r" | "\r\n" }
TOKEN: { <PLUS: "+" > }
TOKEN: { <NUMBER: ( [ "0" - "9" ] + ) > }
```

```
void Start():
{
{
    <NUMBER>
    (
        <PLUS>
        <NUMBER>
    ) *
    <EOF>
```

```
}
```

4.3 Adder (Parser does the addition)

```
PARSER_BEGIN(Adder)
class Adder{
    static void main(String args[]) throws ParseException,
TokenMgrError
    {
        Adder parser=new Adder(System.in);
        int val = parser.Start() ;
        System.out.println(val);
    }
}
PARSER_END(Adder)
```

```
SKIP:{" "}
SKIP:{"\n"|\r"|\r\n"}
TOKEN:{"<PLUS:+">}
TOKEN:{"<NUMBER:([ "0"- "9"])+>}
```

```
int Start() throws NumberFormatException :
{
    Token t ;
    int i ;
    int value ;
}
{
    t = <NUMBER>
    { i = Integer.parseInt( t.image ) ; }
    { value = i ; }
    (
        <PLUS>
        t = <NUMBER>
        { i = Integer.parseInt( t.image ) ; }
        { value += i ; }
    )*
    <EOF>
    { return value ; }
```

5. Project 1 Due 11:59pm Friday 4th February

<http://www.cs.purdue.edu/homes/hosking/352/project/project1.html>

For this project you will implement an interpreter for the *straight-line programming language* described in Chapter 1 of the textbook

Look at Hashtable:

<http://java.sun.com/j2se/1.4.2/docs/api/java/util/Hashtable.html>

References:

1. <http://dinosaur.compilertools.net/yacc/>
2. <http://www.engr.mun.ca/~theo/JavaCC-FAQ/javacc-faq.htm>
3. <http://www.engr.mun.ca/~theo/JavaCC-Tutorial/javacc-tutorial.pdf>