CS 460

Programming Languages

Fall 2023

Dr. Watts

(15 November 2023)

Assignments

- Exercise 2
  - Script running so that your groups can improve their testing techniques
- Exercise 3
  - Your secret folder now contains the .dbg, .lst, .p1, and .p2 files for your lastnameE3.pl460 submission
- Exercise 4
  - .txt file due Wednesday, 2:30 pm
  - We will discuss this in class

Project 2

- Spec and Framework posted
- Extra Credit
  - PL460 program that uses all 93 grammar rules
  - Thursday, 16 November 2023, 6:59 am
  - Draw from Exercise 3
- Suggestions
  - Create First and Follow sets
  - Start with sets for the “Short Grammar”
  - Add in the remaining grammar rules
  - Testing

Exercise 4 Preliminary

- Why?

Expressions and Assignment Statements (Chapter 7)

- Arithmetic Expressions
- Overloaded Operators
- Type Conversions
- Relational and Boolean Expressions
- Short-Circuit Evaluation
- Assignment Statements
- Mixed-Mode Assignment

Arithmetic Expressions

- Operators
- Operator Evaluation Order
  - Precedence
  - Commutativity
  - Associativity
  - Parenthesis
  - Conditional Expressions
  - Operand Evaluation Order
    - Side Effects

Overloaded Operators – Ex 5

- `money operator + (const money & M) const;`
- `money operator += (const money & M);`
- `money operator - (const money & M) const;`
- `money operator -= (const money & M);`
- `money operator * (const double & F) const;`
- **`friend money operator * (const double & Factor, const money & M);`**
- `money operator *= (const double & Factor);`
- `money operator / (const double & Divisor) const;`
- `money operator /= (const double & Divisor);`
- `money operator % (const int & Divisor) const;`
- `money operator %= (const int & Divisor);`
- `money operator ++ (); // Pre increment`
- `money operator ++ (int); // Post increment`
- `money operator -- (); // Pre decrement`
- `money operator -- (int); // Post decrement`

- `bool operator == (const money & M) const;`
- `bool operator != (const money & M) const;`
- `bool operator < (const money & M) const;`
- `bool operator <= (const money & M) const;`
- `bool operator > (const money & M) const;`
- `bool operator >= (const money & M) const;`

Overloaded Operators – Ex 5

- `friend istream & >> (istream & ins, money & M);`

- `friend ostream & << (ostream & outs, const money & M);`

Overloaded Operators – Ex 5

- How do these differ?
  - `money operator * (const double & F) const;`
  - **`friend money operator * (const double & Factor, const money & M);`**
  - `money operator *= (const double & Factor);`

Overloaded Operators – Ex 5

- How do these differ?
  - `money operator ++ (); // Pre increment`
  - `money operator ++ (int); // Post increment`
  - `money operator -- (); // Pre decrement`
  - `money operator -- (int); // Post decrement`