Computer Science
Engineering School

Software
Engineering

Lab 11
Code Generation I

Francisco Ortín Soler

University of Oviedo

Objective

- Generate code for (in that order)
  - Expressions:
    - Integer, character and real constants
    - Variables
    - Arithmetic, comparison and logical expressions
    - Casts
  - Statements: read, write and assignment
  - Definition of global and local variables
  - Function definitions
  - Program

Question

- Name the **necessary code functions** for the following syntactic constructs :
  - Expressions:
    - Integer, character and real constants
    - Variables
    - Arithmetic, comparison and logical expressions
    - Casts
  - Statements: read, write and assignment
  - Definition of global and local variables
  - Function definitions
  - Program

Question

- Define the following code template

value[[Logical: $expression_1$ ⟶
$expression_2$ (&& | ||) $expression_3$]]=

Activity 1

- Write your **code templates** as multiline comments at the beginning of
  a. AddressCGVisitor.java for Address templates
  b. ValueCGVisitor.java for Value templates
  c. ExecuteCGVisitor.java for Execute templates
  - Be careful with the productions of the abstract grammar (they cannot be wrong)
  - Show the AG to the lecturer before its implementation (mandatory)

Implementation

- Important: Pass the name of the output text (MAPL) file to your compiler (i.e., a new argument to the `main` method)
  - Do **not** write code in a given text file (e.g., output.txt)

Questions

1. How many visitors are we going to implement?
2. What are their names?
3. Is there a default behavior?
4. Which one is it?
5. Where are we going to provide such a default behavior?
6. What is going to be the implementation of each visit method?

Implementation

- The following code is low level (writes text in a file):

```
switch (expression1.operator) {
        case "&&": <and>
                   break;
        case "||": <or>
                   break;
}
```

- You'd better move it to a **cg** object replace it with

```
cg.logical(expression1.operator);
```

- Its purpose is just writing code in an output text file

CodeGenerator
- outputFile: FileWriter + logical(operator:String) + push(int) + load(type: Type) …

In this way,
visitors just traverse,
while CG generates code

Activity 2

- You must have your code templates validated by the lecturer before the implementation
- **Implement** the code templates in your C-- compiler
  - At least test it with input.txt file provided
  - **Always run the generated code with MAPL**
    1. Making sure the execution is the expected one
    2. No errors or warnings are shown