編譯器設計

Project 2
Scala⁻ Syntactic Definition

Scala⁻ : A Simple Scala Language

- Project 2
  - Use *yacc* to specify the syntactic definition
- Deadline
  - 1:20pm, June 9 (Tuesday)
- What to submit
  - Revised lex scanner
  - New yacc parser
  - Makefile
  - A note
  - Test programs, if necessary

Data Types and Declarations

- Predefined data types
  - Scalar: char, string, int, boolean, and float
- Variables and constants
  - Global variables
    - declared outside methods
  - Local variables
    - declared inside methods

Data Types and Declarations

- Constant Declaration

  val *identifier* <: *type* > **=** *constant_exp*

  - Constants can not be reassigned
- Examples

```
val s = "Hey There"
val i = -25
val f = 3.14
val b:boolean = true
```

Data Types and Declarations

Variable declaration

var *identifier* <: *type;*>< **=** *constant_exp* >

Examples

```
var s : string
var i = 10
var d : real
var b: boolean = false
```


Data Types and Declarations

Array declaration

var *identifier* **:** *type* **[** *num* **]**

Examples

```
var a: int [10]          // an array of 10 integer elements
var b: boolean [5]       // an array of 6 boolean elements
var f: float [100]       // an array of 100 float elements
```

Program Units

Program

object *identifier* {
<zero or more variable and constant declarations>
one or more method declarations
}

- < > : optional
- Every program has at least one method: main

Program Units

Methods

- A method is declared with the format

  def *identifier* (<formal arguments >) <: *type* >
  {
  <zero or more constant and variable declarations>
  <zero or more statements>
  }

- Parameters
  - a list of declarations separated by comma

  identifier : *type* <, *identifier* : *type*, ... , *identifier* : *type*>
- Type
  - a predefined data type

Program Units

Procedures

```
object example {
  // constants and variables
  val a = 5
  var c : int
  // procedure declaration
  def add(a:int, b:int) : int
  {
    return a+b
  }

  // main statements
  def main ( )
  {
    c = add(a, 10)
    println (c)
  }
}
```


Statements

Simple

identifier = *expression*

or

identifier[*integer_expression*] = *expression*

or

print (*expression*) or **println** (*expression*)

or

read *identifier*

or

return or **return** *expression*

Expressions

(1) – (unary)
(2) * /
(3) + –
(4) < <= == => > !=
(5) !
(6) &&
(7) ||

Statements

- Block
  - A list of statements enclosed by { and }

{
<zero or more variable and constant declarations>
<one or more statements>
}

Statements

- Conditionals

if (*boolean_expr*)
a block or simple statement
else
a block or simple statement

or

if (*boolean_expr*)
a block or simple statement

Statements

- Loops

  while (*boolean_expr*)
  a block or simple statements

  or

  for (*identifier* <**-** *num* **to** *num*)
  a block or simple statement

- Procedure invocations

  identifier <(comma-separated expressions)>

Semantic Definitions

- Call-by-value
- Scope rules
- Types of assignments
- Types of parameters

Example

```
/*
 * Example with Functions
 */

object example {
  val a = 5

  // function declaration
  def add (a: int, b: int) : int
  {
    return a+b
  }

  // main statements
  def main()
  {
    var c:int
    c = add(a, 10)
    if (c > 10)
      print -c
    else
      print c
    println ("Hello World")
  }
}
```