OBLIG 1 INF 5110

Andreas Heim (andrheim)

Gjennomføring

Semantisk analyse

Nodene har selv ansvar for å sjekke semantikken og legge til seg selv i symboltabellen. For å håndtere konvertering fra int til float introduserte jeg en funksjon i Type-klassene, Type.canBeCastTo(otherType). Selv om denne i praksis bare gjelder mellom int og float, så føler jeg at koden blir litt ryddigere for hver “instanceof”-expression som jeg slipper å skrive. I alle henseender innenfor kompilatoren vil en int være likeverdig en float, så jeg kunne sannsynligvis ha gjort slik at Int arvet float.

Symboltabell

Symboltabellen er stackbasert og legger en ny HashMap<Name, Type> på stacken hver gang man går inn i et nytt skop. Tabellen tar ikke vare på “gamle” skop.

Siden grunnstrukturen i symboltabellen er en hashmap, så viste det seg problematisk å håndtere metodesignatur. Jeg følte ikke at metodesignatur og returtype hang sammen, så jeg introduserte en ny klasse i syntakstreet, FunctionName. Denne klassen tar imot formelle parametre i tillegg til funksjonsnavnet.

I CallStatement blir det laget en instans av FunctionName på bakgrunn av aktuelle parametere. I FunctionName-klassen har jeg overskrevet equals-metoden til å sammenlikne, navn, parameter-rekkefølge/type og om parameteren er pass-by-value/reference, slik at lookup-funksjonen i HashMap fungerer som ønsket.

Bytekode

Alle standardbibliotek blir lagt til symboltabellen og kodegenerert i Program-klassen (øverste nivå).

Alle Type-klassene har en metode “getByteCodeType()” som returnerer korresponderende type for bytecode-biblioteket. Unntaket er ClassType som trenger oppslag i kodefilen for å finne strukt-id. Dette blir sjekket der hvor variable blir deklarert i syntakstreet, men denne logikken kunne også med fordel vært flyttet inn i ClassType eller en Util-klasse.

I de aritmetiske og logiske uttrykksklassene genererer selve operanden sin egen kode, dette gjør kodegenereringsmetodene små og lettleste.

I AssignStatement-klassen er det lagt inn sjekk for om variabelen er lokal eller global. Dvs. om man finner en lokal variabel brukes denne, hvis ikke leter man etter (og bruker) en global. Denne sjekken kunne blitt enklere hvis symboltabellen hadde blitt tatt vare på etter semantisk analyse.

Kjøreinstruksjoner

Det er ikke endret på byggesystem, men det er lagt til ett sett med enhetstester i test-mappen. Disse kan kjøres via eclipse/intellij.

Kjente feil & Mangler

I AndOp og OrOpExpression er det ikke lagt inn short-circuit evaluering slik som beskrevet i språknotatet.

Mangelfull output fra parser/semantisk analyse om hvor feilen befinner seg.

Test 27 (test_nested_scopes.d) feiler ved generering av kode. Tester er kjørt med utkommentert “generateCode”.

Utskrift fra ant test.

```
test:
     [java] Testing compiler class.
     [java] Testing files in /Users/andreas heim/code/skole/kompilatorteknikk-oblig1/tests
     [java] Test[1] SHOULD FAIL: assign_left_side_not_exist_fail.d
     [java] Error: Symbol not declared. VariableExpression - var b
     [java] Test[2] SHOULD FAIL: function1_fail.d
     [java] expressionType = void
     [java] Error: Incompatible return typeCallStatement. Calling Func
     [java] Test[3] SHOULD FAIL: function2_fail.d
     [java] Error: Incompatible return
typedflat.syntaxtree.statement.ReturnStatement@7e79b177
     [java] Test[4] SHOULD FAIL: function3_fail.d
     [java] Error: Incompatible return
typedflat.syntaxtree.statement.ReturnStatement@67723c7f
     [java] Test[5] SHOULD FAIL: function4_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling Add
     [java] Test[6] SHOULD FAIL: function5_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling Add
     [java] Test[7] SHOULD FAIL: function_calls_not_function_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling Test
     [java] Test[8] SHOULD FAIL: function_does_not_exist_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling yy
     [java] Test[9] SHOULD FAIL: function_name_used_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling print_str
     [java] Test[10] SHOULD FAIL: function_no_return_fail.d
     [java] Error: Function must have a return
statement.dflat.syntaxtree.decl.FuncDecl@115af049
     [java] Test[11] SHOULD PASS: function_return_null.d
     [java] P
     [java] P
     [java] Error: null
     [java] Test[12] SHOULD FAIL: function_unknown_returntype_fail.d
     [java] Error: Type not declared.dflat.syntaxtree.type.ClassType@11c19919
     [java] Test[13] SHOULD FAIL: general1_fail.d
     [java] Error: Symbol already declared. Tmp
     [java] Test[14] SHOULD FAIL: general2_fail.d
     [java] Error: Symbol already declared. a
     [java] Test[15] SHOULD FAIL: general3_fail.d
     [java] Error: Symbol not declared. VariableExpression - var bar
     [java] Test[16] SHOULD FAIL: general4_fail.d
     [java] Error: Main method declared with parameters and/or return type, or not declared
at all.
     [java] Test[17] SHOULD FAIL: general5_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling Func
     [java] Test[18] SHOULD FAIL: general6_fail.d
     [java] Error: Symbol not declared. CallStatement. Calling Func
     [java] Test[19] SHOULD FAIL: main_wrong_parameters_fail.d
     [java] Error: Main method declared with parameters and/or return type, or not declared
at all.dflat.syntaxtree.decl.FuncDecl@24bf1f20
     [java] Test[20] SHOULD FAIL: main_wrong_type_fail.d
     [java] Error: Main method declared with parameters and/or return type, or not declared
at all.dflat.syntaxtree.decl.FuncDecl@7f9480b8
     [java] Test[21] SHOULD FAIL: nested_class_shadow_fail.d
     [java] Error: Symbol not declared. VariableExpression - var outerScope
     [java] Test[22] SHOULD FAIL: return_contradicting_expression_fail.d
```

```
    [java] Error: Incompatible return
typedflat.syntaxtree.statement.ReturnStatement@61e090ee
    [java] Test[23] SHOULD PASS: test1.d
    [java] P
    [java] Error: null
    [java] Test[24] SHOULD PASS: test3.d
    [java] P
    [java] P
    [java] Error: null
    [java] Test[25] SHOULD PASS: test4.d
    [java] P
    [java] P
    [java] Error: null
    [java] Test[26] SHOULD PASS: test5.d
    [java] P
    [java] P
    [java] Error: null
    [java] Test[27] SHOULD PASS: test_nested_scopes.d
    [java] P
    [java] Error: null
    [java] Test[28] SHOULD FAIL: type1_fail.d
    [java] expressionType = int
    [java] Error: Incompatible return
typedflat.syntaxtree.expression.literal.IntLiteral@131f139b
    [java] Test[29] SHOULD FAIL: type2_fail.d
    [java] Error: Incompatible Typedflat.syntaxtree.expression.literal.IntLiteral@1b609c13
    [java] Test[30] SHOULD FAIL: type3_fail.d
    [java] expressionType = float
    [java] Error: Incompatible return
typedflat.syntaxtree.expression.AritOpExpression@63a6b16f
    [java] Test[31] SHOULD FAIL: type4_fail.d
    [java] Error: Incompatible Typedflat.syntaxtree.expression.literal.FloatLiteral@3927ff0d
    [java] Test[32] SHOULD FAIL: type5_fail.d
    [java] Error: Incompatible return typeVariableExpression - var tmp
    [java] Test[33] SHOULD FAIL: type6_fail.d
    [java] Error: Incompatible return
typedflat.syntaxtree.expression.literal.IntLiteral@70d9cbcb
    [java] Test[34] SHOULD FAIL: type7_fail.d
    [java] Error: Symbol not declared. VariableExpression - var DoesntExist
    [java] Test[35] SHOULD FAIL: undeclared_class_fail.d
    [java] Error: Symbol not declared. VariableExpression - var zot
    [java] Test[36] SHOULD FAIL: undeclared_class_var_fail.d
    [java] Error: Symbol not declared. VariableExpression - var zot
    [java] Test[37] SHOULD FAIL: undeclared_new_argument_fail.d
    [java] Error: Incompatible return
typedflat.syntaxtree.statement.ReturnStatement@6ad2b64e
    [java] Test[38] SHOULD FAIL: variable_scope_shadow_fail.d
    [java] Error: Incompatible
Typedflat.syntaxtree.expression.NegatedExpression@2fbef1ac
    [java] Test[39] SHOULD FAIL: wrong_arith_operand_type_fail.d
    [java] Error: Incompatible
Typedflat.syntaxtree.expression.AritOpExpression@20ec6bb1
    [java] Test[40] SHOULD FAIL: wrong_log_exp_type_fail.d
    [java] Error: Incompatible Typedflat.syntaxtree.expression.OrOpExpression@75c45731
    [java] Test[41] SHOULD FAIL: wrong_new_arg_type_fail.d
    [java] Error: Incompatible return
typedflat.syntaxtree.statement.ReturnStatement@2d388e5e
    [java] Test[42] SHOULD FAIL: wrong_not_operand_type_fail.d
    [java] Error: Incompatible
Typedflat.syntaxtree.expression.NegatedExpression@5c0ad483
```

[java] Test[43] SHOULD FAIL: wrong_rel_operand_type_fail.d
[java] Error: Incompatible Typedflat.syntaxtree.expression.RelOpExpression@1b275a34
[java] Test completed successfully.

Utskrift fra ant list-runme (med modifikasjoner på read_int, siden det feiler ved kjøring med ant)

```
[java] Loading from file: code-examples/RunMe.bin
[java] Variables:
[java] 0: var Complex dummy
[java] Procedures:
[java] 0: func void print_float(float 0)
[java] 1: func void print_int(int 0)
[java] 2: func void print_line(string 0)
[java] 3: func void print_str(string 0)
[java] 4: func float read_float()
[java] 5: func int read_int()
[java] 6: func int read_char()
[java] 7: func string read_string()
[java] 8: func Complex Add(Complex 0, Complex 1)
[java]    var Complex 2
[java]    0: new Complex
[java]    3: storelocal 2
[java]    6: loadlocal 0
[java]    9: getfield Complex[0] {float}
[java]    14: loadlocal 1
[java]    17: getfield Complex[0] {float}
[java]    22: add
[java]    23: loadlocal 2
[java]    26: putfield Complex[0] {float}
[java]    31: loadlocal 0
[java]    34: getfield Complex[1] {float}
[java]    39: loadlocal 1
[java]    42: getfield Complex[1] {float}
[java]    47: add
[java]    48: loadlocal 2
[java]    51: putfield Complex[1] {float}
[java]    56: loadlocal 2
[java]    59: return
[java] 9: func int Max(int 0, int 1)
[java]    var int 2
[java]    0: loadlocal 0
[java]    3: loadlocal 1
[java]    6: gt
[java]    7: jmpfalse 19
[java]    10: loadlocal 0
[java]    13: storelocal 2
[java]    16: jmp 26
[java]    19: nop
[java]    20: loadlocal 1
[java]    23: storelocal 2
[java]    26: nop
[java]    27: loadlocal 2
[java]    30: return
[java] 10: func void printCmplx(Complex 0)
[java]    0: pushstring "Real "
[java]    3: call print_str {3}
```

```
[java]     6: loadlocal 0
[java]     9: getfield Complex[0] {float}
[java]     14: call print_float {0}
[java]     17: pushstring ""
[java]     20: call print_line {2}
[java]     23: pushstring "Imag "
[java]     26: call print_str {3}
[java]     29: loadlocal 0
[java]     32: getfield Complex[1] {float}
[java]     37: call print_float {0}
[java]     40: pushstring ""
[java]     43: call print_line {2}
[java]     46: return
[java] 11: func void test()
[java]     var Complex 0
[java]     var Complex 1
[java]     var Complex 2
[java]     var int 3
[java]     var int 4
[java]     var int 5
[java]     0: new Complex
[java]     3: storelocal 0
[java]     6: new Complex
[java]     9: storelocal 1
[java]     12: pushint 1
[java]     17: loadlocal 0
[java]     20: putfield Complex[0] {float}
[java]     25: pushint 2
[java]     30: loadlocal 0
[java]     33: putfield Complex[1] {float}
[java]     38: pushint 3
[java]     43: loadlocal 1
[java]     46: putfield Complex[0] {float}
[java]     51: pushint 4
[java]     56: loadlocal 1
[java]     59: putfield Complex[1] {float}
[java]     64: loadlocal 0
[java]     67: loadlocal 1
[java]     70: call Add {8}
[java]     73: call printCmplx {10}
[java]     76: pushint 3
[java]     81: storelocal 3
[java]     84: pushint 7
[java]     89: storelocal 4
[java]     92: loadlocal 4
[java]     95: loadlocal 3
[java]     98: call Max {9}
[java]     101: storelocal 5
[java]     104: pushstring "Max in test"
[java]     107: call print_line {2}
[java]     110: loadlocal 5
[java]     113: call print_int {1}
[java]     116: pushstring ""
[java]     119: call print_line {2}
[java]     122: return
[java] 12: func void printStr(string 0)
[java]     0: loadlocal 0
[java]     3: call print_str {3}
[java]     6: return
[java] 13: func void inOutTest()
```

```
[java]     var int 0
[java]     var int 1
[java]     0: pushstring "skriv v1 (12)"
[java]     3: call print_line {2}
[java]     6: pushint 12
[java]     11: storelocal 0
[java]     14: pushstring "skriv v2 (21)"
[java]     17: call print_line {2}
[java]     20: pushint 21
[java]     25: storelocal 1
[java]     28: pushstring "Storst "
[java]     31: call print_str {3}
[java]     34: loadlocal 0
[java]     37: loadlocal 1
[java]     40: call Max {9}
[java]     43: call print_int {1}
[java]     46: pushstring ""
[java]     49: call print_line {2}
[java]     52: return
[java] 14: func void Main()
[java]     var float 0
[java]     var int 1
[java]     var string 2
[java]     0: pushfloat 6.48074
[java]     5: storelocal 0
[java]     8: loadlocal 0
[java]     11: call print_float {0}
[java]     14: pushstring ""
[java]     17: call print_line {2}
[java]     20: pushint 7
[java]     25: storelocal 1
[java]     28: loadlocal 1
[java]     31: call print_int {1}
[java]     34: pushstring ""
[java]     37: call print_line {2}
[java]     40: pushstring "TestNavn"
[java]     43: storelocal 2
[java]     46: loadlocal 2
[java]     49: call printStr {12}
[java]     52: pushstring ""
[java]     55: call print_line {2}
[java]     58: call test {11}
[java]     61: call inOutTest {13}
[java]     64: new Complex
[java]     67: storeglobal dummy{Complex}
[java]     70: pushfloat 1.0
[java]     75: loadglobal dummy
[java]     78: putfield Complex[0] {float}
[java]     83: pushfloat 2.0
[java]     88: loadglobal dummy
[java]     91: putfield Complex[1] {float}
[java]     96: loadglobal dummy
[java]     99: call printCmplx {10}
[java]     102: pushstring "DONE"
[java]     105: call print_line {2}
[java]     108: return
[java] Structs:
[java] 0: Complex
[java]     0: float
[java]     1: float
```

[java] Constants:
[java] 0: Real
[java] 1:
[java] 2: Imag
[java] 3:
[java] 4: Max in test
[java] 5:
[java] 6: skriv v1 (12)
[java] 7: skriv v2 (21)
[java] 8: Storst
[java] 9:
[java] 10:
[java] 11:
[java] 12: TestNavn
[java] 13:
[java] 14: DONE
[java] STARTWITH: Main