CS 241 Tutorial 11

Graham Cooper

July 24th, 2015

Extending WLP4

Adding a Boolean type ot WLP4

should be allowed to:

- declare bool vars, initialized to true or false
- assign to bool vars
- write expressions with !, &&, || that take booleans as operands
- write test expressions using the comparison operators to evaluate boolean values
- use bool vars or expresions as test conditions for if and while statements
- will not allow:
  - pointer to bool
  - arrays of bool
  - bool as parameter of wain
  - printing bool with println
  - returning bool from a procedure
  - arithmetic involving int/int*
  - boolean expresions involving int/int* and bool
  - $<, >, <=, >=$, between bools
  - using 0 and 1 to mean false and true

```
int wain(int x, int y){
bool a = true;
bool b = false;
a = x > y;
if(a){
-- println(x);
}
```

```
else {
-- println(y);
}
b = true;
while(b || a) && (x < y)){
-- if (true){
-- -- b = !((a || b) && a);
-- } else {}
}
return y;
}
```

Lexical Syntax

bool
true
false
true and false become a BVAL

dcls − > dcls dcl BECOMES NUM SEMI
dcls − > dcls dcl BECOMES NULL SEMI

Context free syntax

statement − >
type − > BOOL
dcls − > dcls dcl BE ... BVAL SEMI
factor − > BVAL
expr − > bexpr || bterm
term − > term && factor
factor − > ! factor
test − > bexpr
test − > test == bexpr
btest − > btest < bexpr

Context-Sensitive Syntax

- type rules
- sybol table stuff

```
code(factor1 -> !factor2) =
code(factor2)
sub $3, $11, $3

code(expr1 -> expr2 || term) =
code(expr2)
beq $3, $11, endOrX
code(term)
endOrX
```