Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Boo-Compiler

A boolean programming language compiler

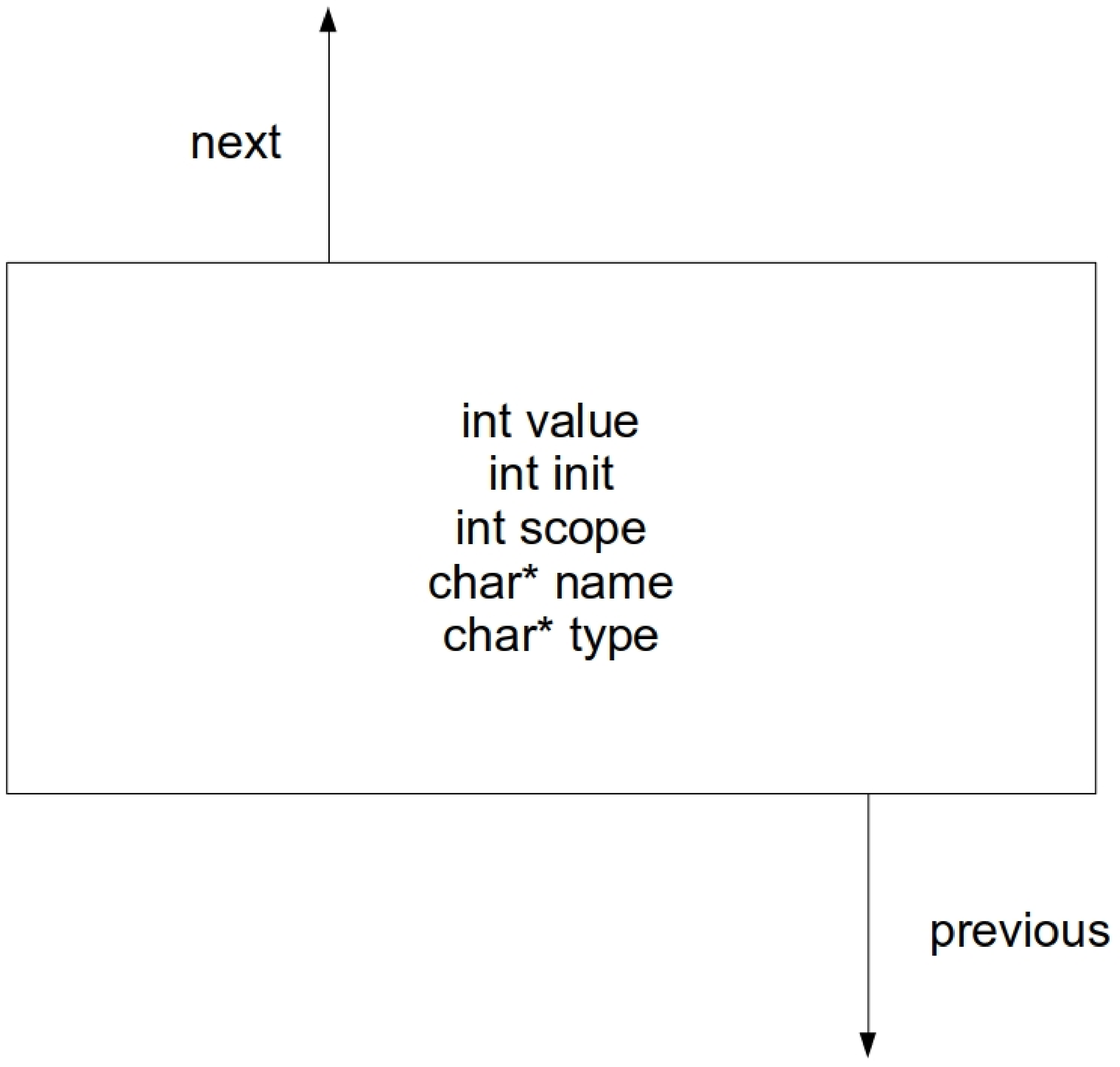
Authors
Andriy Tyyko, Marco Mondini, Luca Sabiucciu

Formal Languages and Compilers
Academic Year 2016/2017 – Winter Session
Faculty of Computer Science, Free University of Bolzano

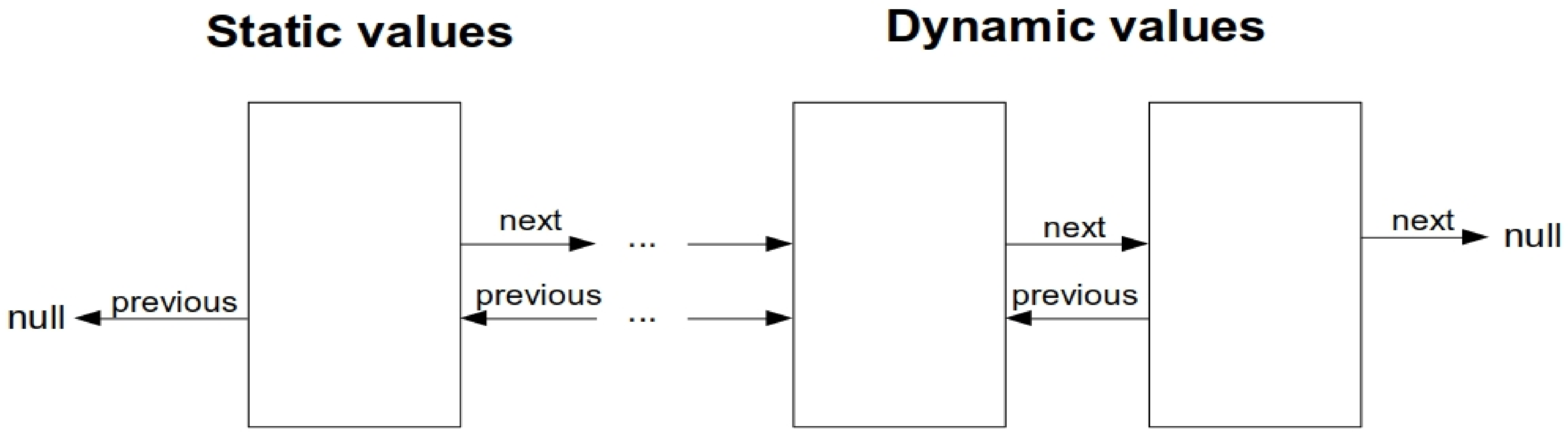
Idea & Features

- Simple boolean programming language
- true/false as well as binary (0/1)
- Simple numerical calculations (+, -, *, / ...)
- Variables of different types
- Print results, variable’s value and strings
- Every instruction is delimited by a dot (‘.’)

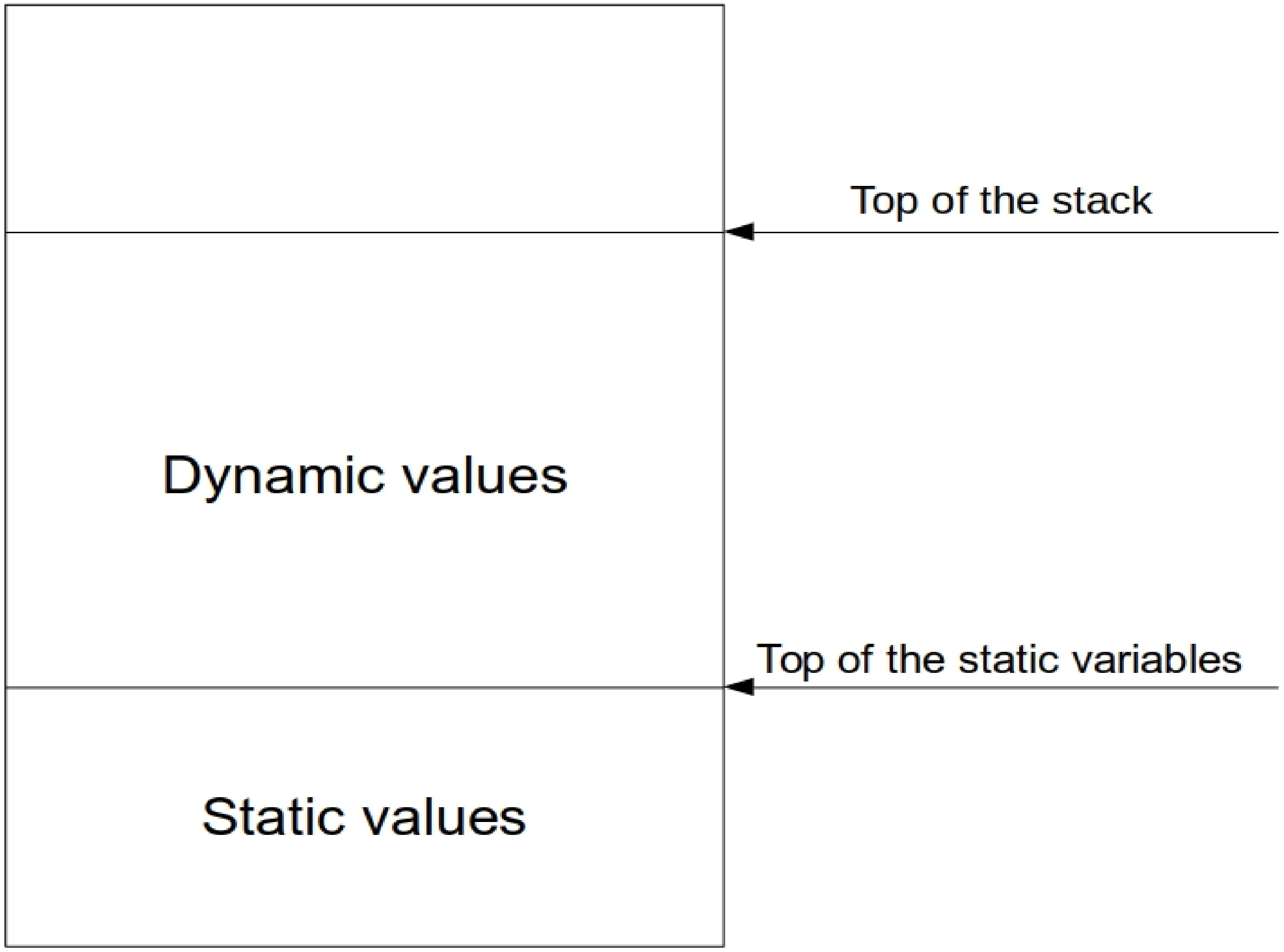
Data


next
int value
int init
int scope
char* name
char* type
previous

Symbol Table



Stack Management



Stack Management (Ex.)

```
bool a=true.
bool b.
{
    int a=0.
    print a.
}
print a.
```

Before starting to read the input program, create a poll into the stack with some frequent values

Current scope = 0

TopOfTheStack

StaticTop

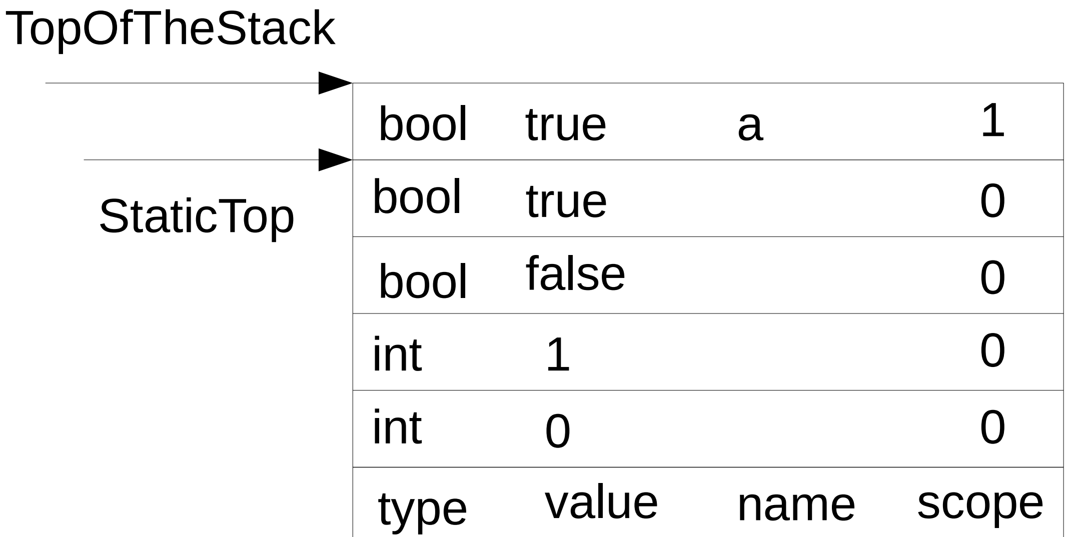
bool	true		0
bool	false		0
int	1		0
int	0		0
type	value	name	scope

Stack Management (Ex.)

```
──►bool a=true.
   bool b.
   {
       int a=0.
       print a.
   }
   print a.
```

Add to the stack the boolean variable named a, with value true

Current scope = 1



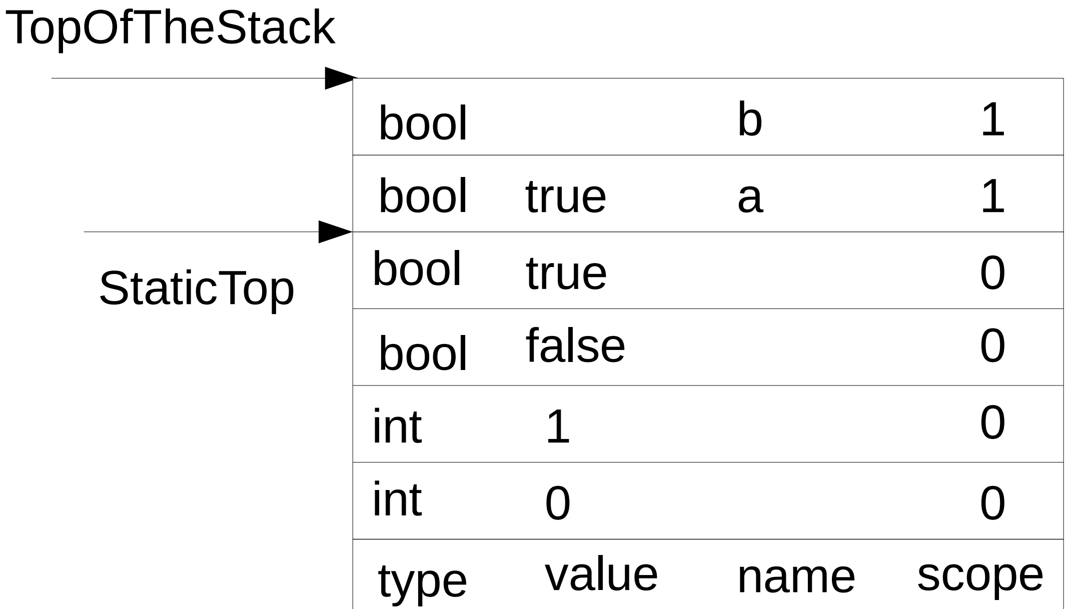
type	value	name	scope
bool	true	a	1
bool	true		0
bool	false		0
int	1		0
int	0		0

Stack Management (Ex.)

```
   bool a=true.
→  bool b.
   {
       int a=0.
       print a.
   }
   print a.
```

Add to the stack the variable b, of type bool. This time it does not have a value

Current scope = 1



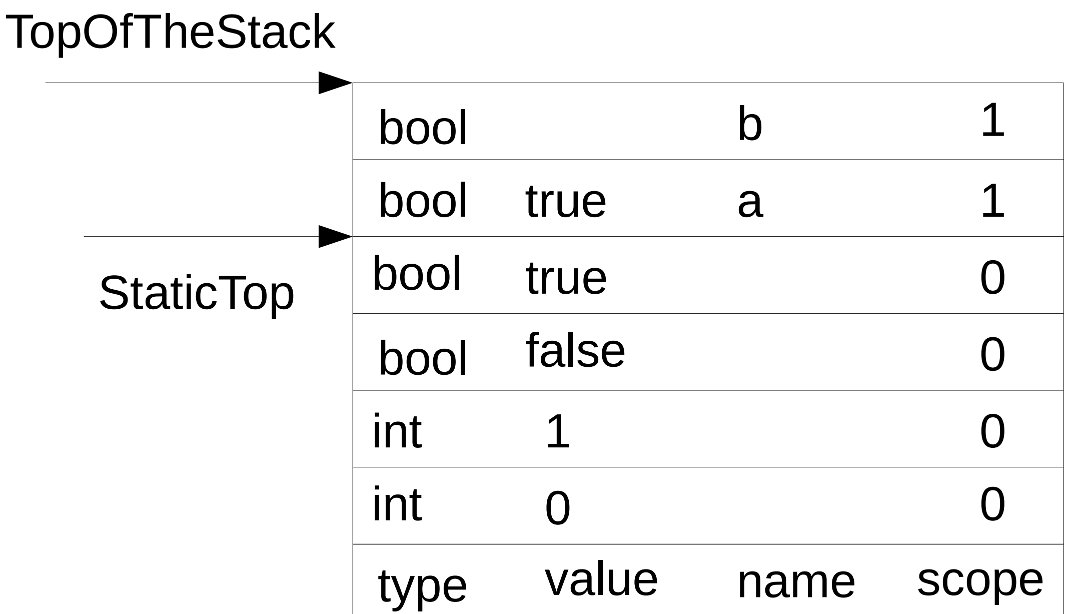
type	value	name	scope
bool		b	1
bool	true	a	1
bool	true		0
bool	false		0
int	1		0
int	0		0

Stack Management (Ex.)

```
bool a=true.
bool b.
{
    int a=0.
    print a.
}
print a.
```

Increase the scope counter by 1.
Curly brackets create a new scope, allowing for variable shadowing

Current scope = 2



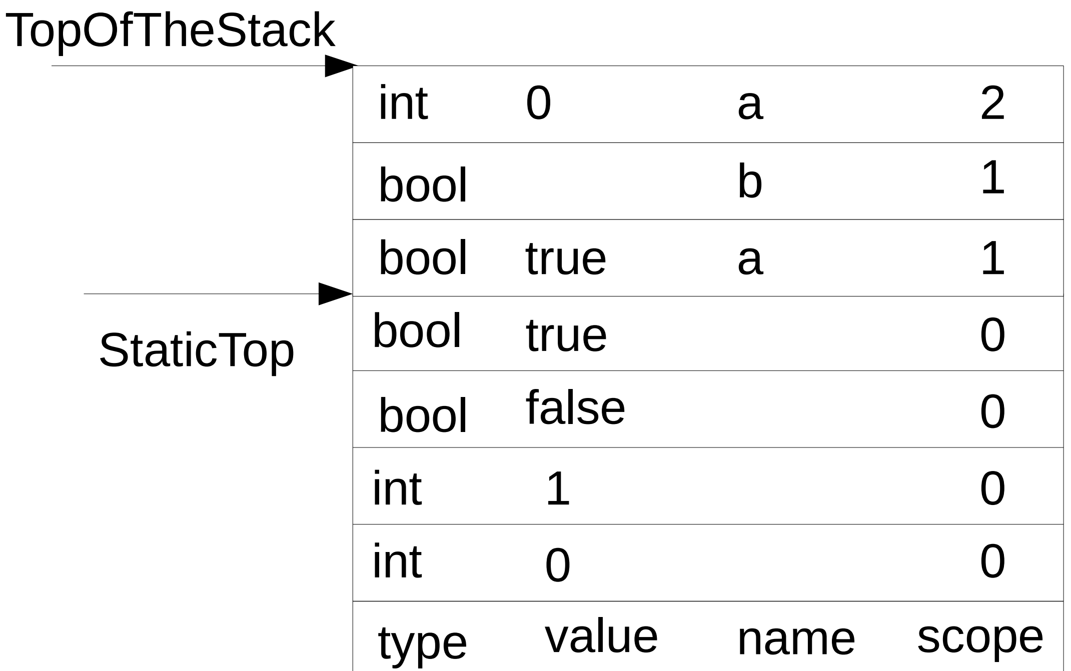
bool		b	1
bool	true	a	1
bool	true		0
bool	false		0
int	1		0
int	0		0
type	value	name	scope

Stack Management (Ex.)

```
  bool a=true.
  bool b.
  {
→     int a=0.
      print a.
  }
  print a.
```

Variable a of type int has been already declared, **but** in a different scope, so add it on top of the stack

Current scope = 2



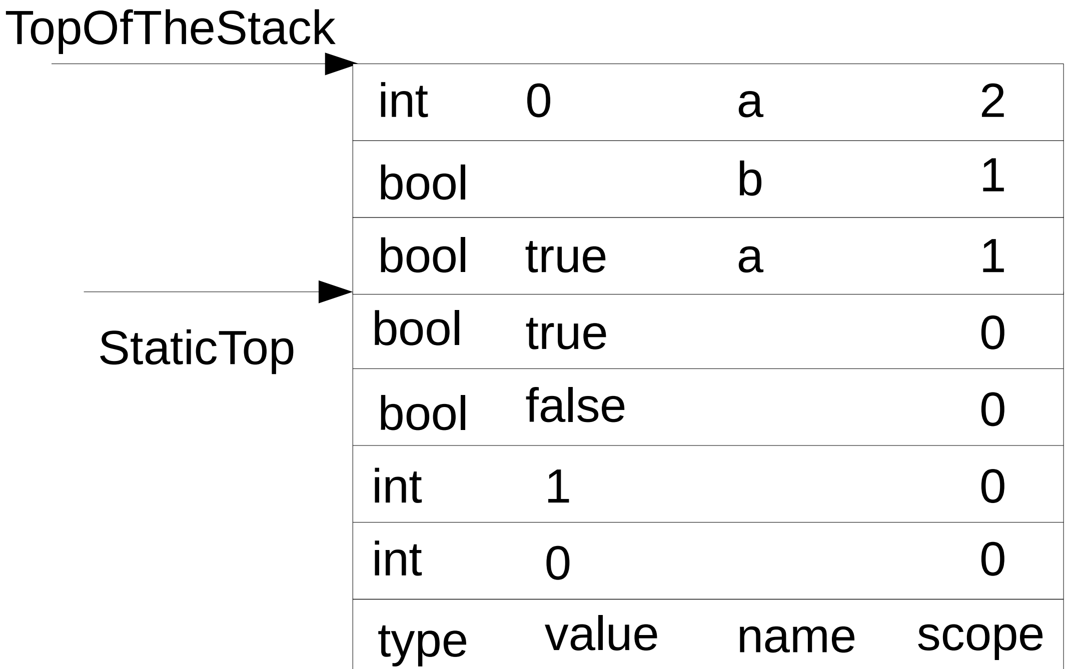
type	value	name	scope
int	0	a	2
bool		b	1
bool	true	a	1
bool	true		0
bool	false		0
int	1		0
int	0		0

Stack Management (Ex.)

```
bool a=true.
bool b.
{
    int a=0.
    print a.
}
print a.
```

The variable a is printed. Since variable shadowing allowed to declare two different a's, the most recent is used (topmost of the stack)

Current scope = 2



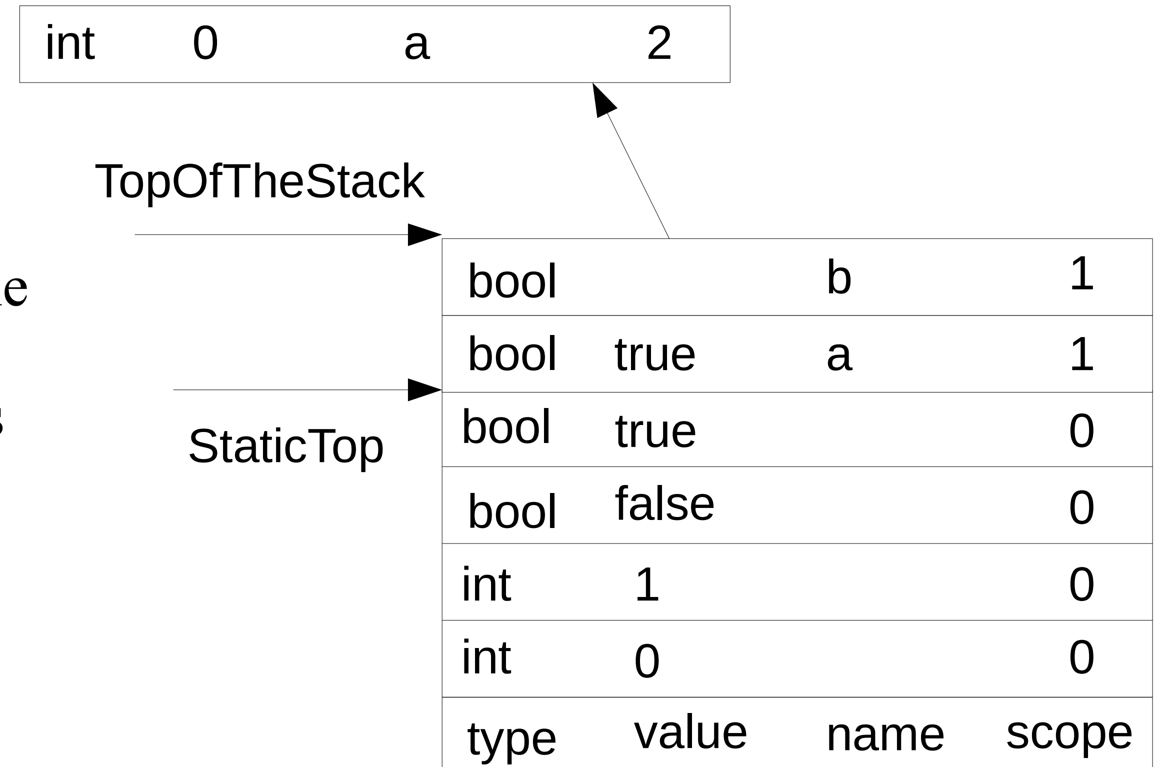
type	value	name	scope
int	0	a	2
bool		b	1
bool	true	a	1
bool	true		0
bool	false		0
int	1		0
int	0		0

Stack Management (Ex.)

```
bool a=true.
bool b.
{
    int a=0.
    print a.
}
print a.
```

A closed curly bracket is encountered, meaning the end of the current scope. The scope counter is decreased by 1 and all the elements belonging to the former scope are removed from the stack

Current scope = 1

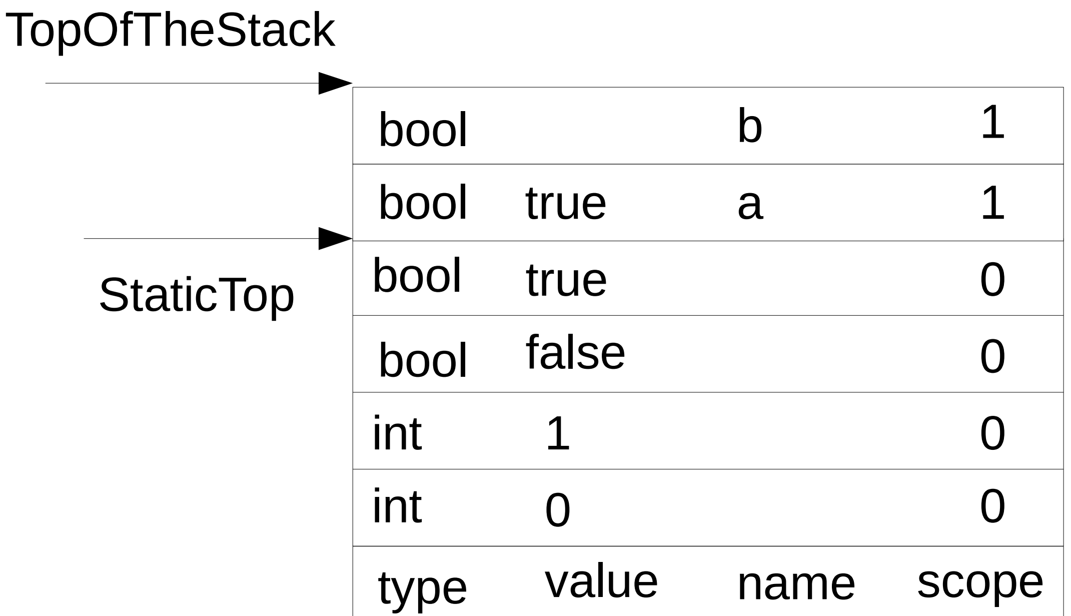


Stack Management (Ex.)

```
bool a=true.
bool b.
{
    int a=0.
    print a.
}
print a.
```

The variable, named a, is printed. This time the first declared, with value true, since it is not shadowed anymore from the one in previous scope. The value printed is true.

Current scope = 1



type	value	name	scope
bool		b	1
bool	true	a	1
bool	true		0
bool	false		0
int	1		0
int	0		0

Expression Management (Ex.)

Print 1+1 and 1.

Compute the value for the expression 1+1and1 before printing it's value.

Current scope = 1

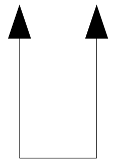
TopOfTheStack

StaticTop

bool	true		0
bool	false		0
int	1		0
int	0		0
type	value	name	scope

Expression Management (Ex.)

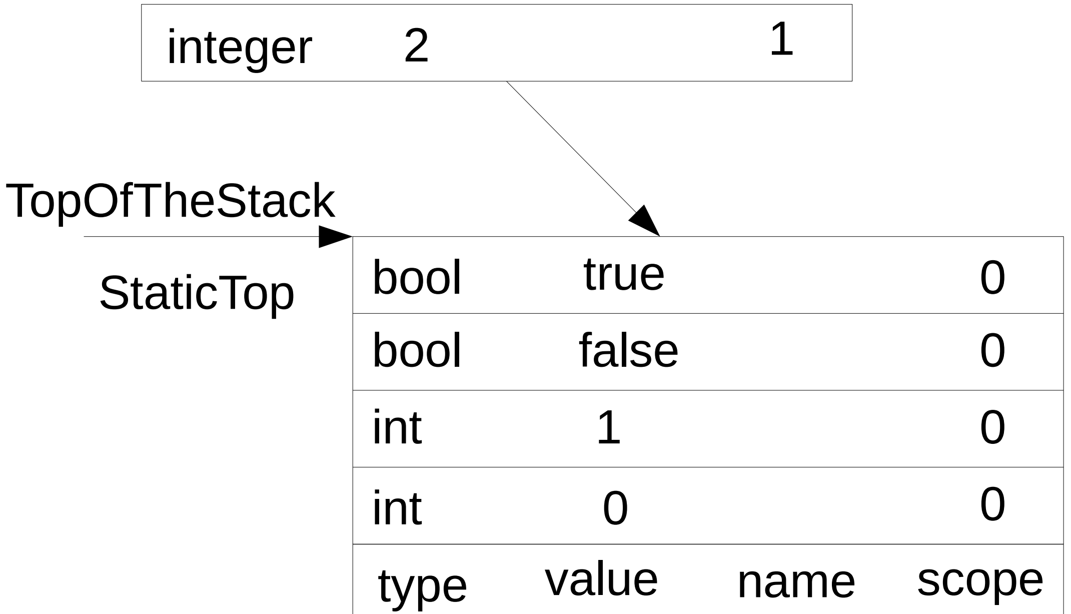
Print 1+1 and 1.



Type → Integer
Value → 2
Scope → 1

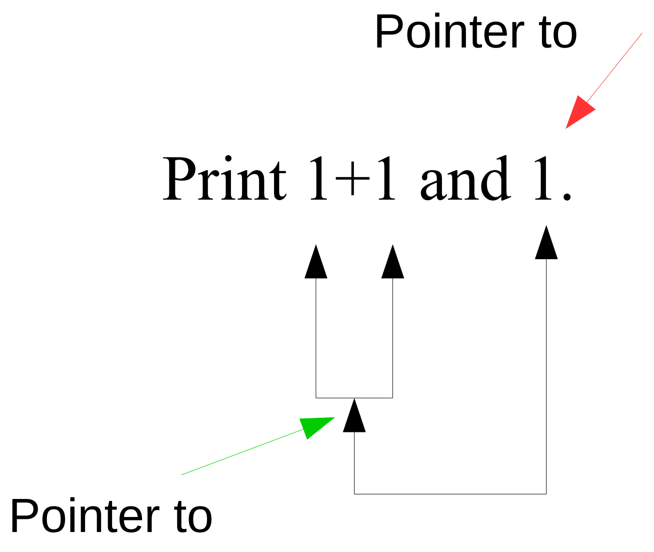
Add a new variable, without a name to the stack

Current scope = 1



type	value	name	scope
bool	true		0
bool	false		0
int	1		0
int	0		0

Expression Management (Ex.)



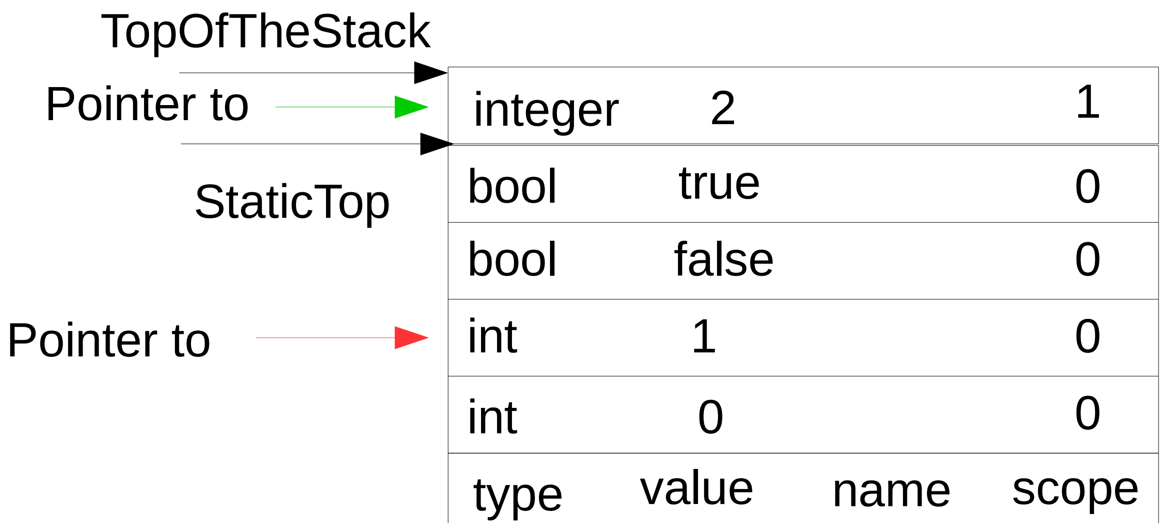
Type → Int
Value → 1
Scope → 1

Operation:
<Integer> AND <int>

Automatic cast integer to int:
If value > 0 then value = 1
else value = 0

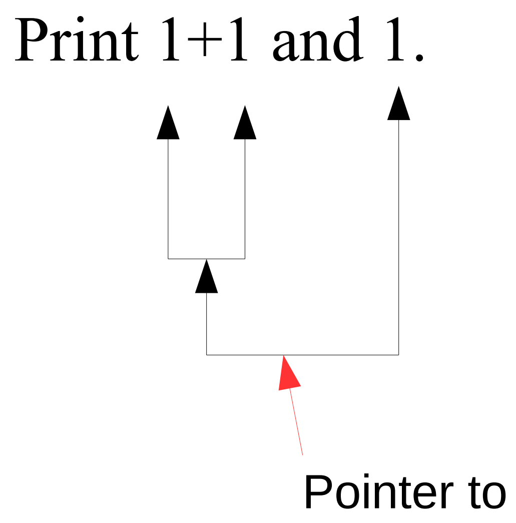
Instead of adding a new element to the stack, reference one of the static ones, saving space for other elements

Current scope = 1



type	value	name	scope
integer	2		1
bool	true		0
bool	false		0
int	1		0
int	0		0

Expression Management (Ex.)

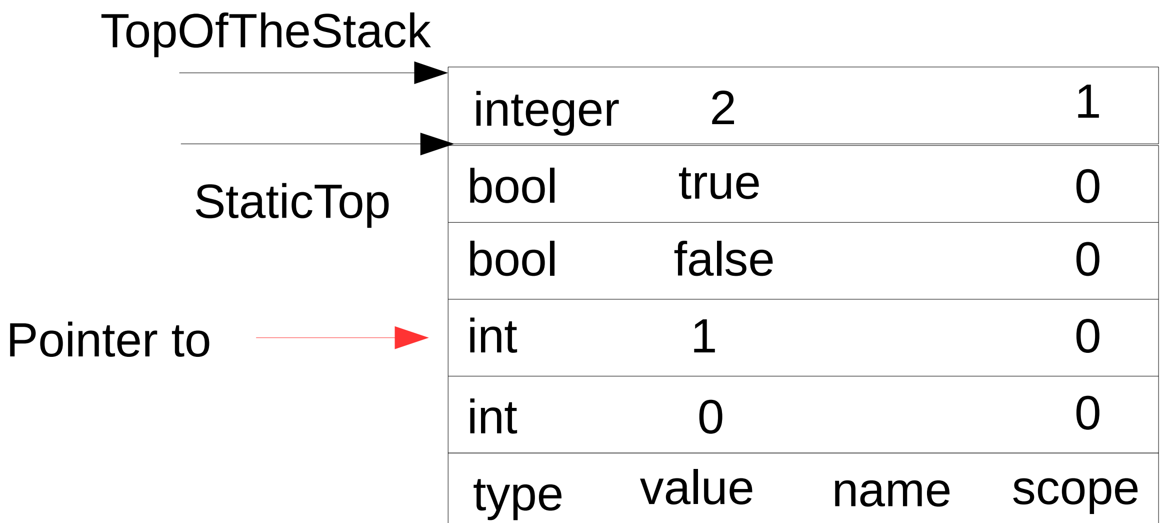


Operation:
<Integer> AND <int>

Automatic cast integer to int:
If value > 0 then value = 1
else value = 0

Prints the value of the element referenced (red arrow), thus 1.

Current scope = 1



type	value	name	scope
integer	2		1
bool	true		0
bool	false		0
int	1		0
int	0		0

Some Stats

136 states
65 productions (+1 for the acceptance state of the augmented grammar)

Thanks for the attention!