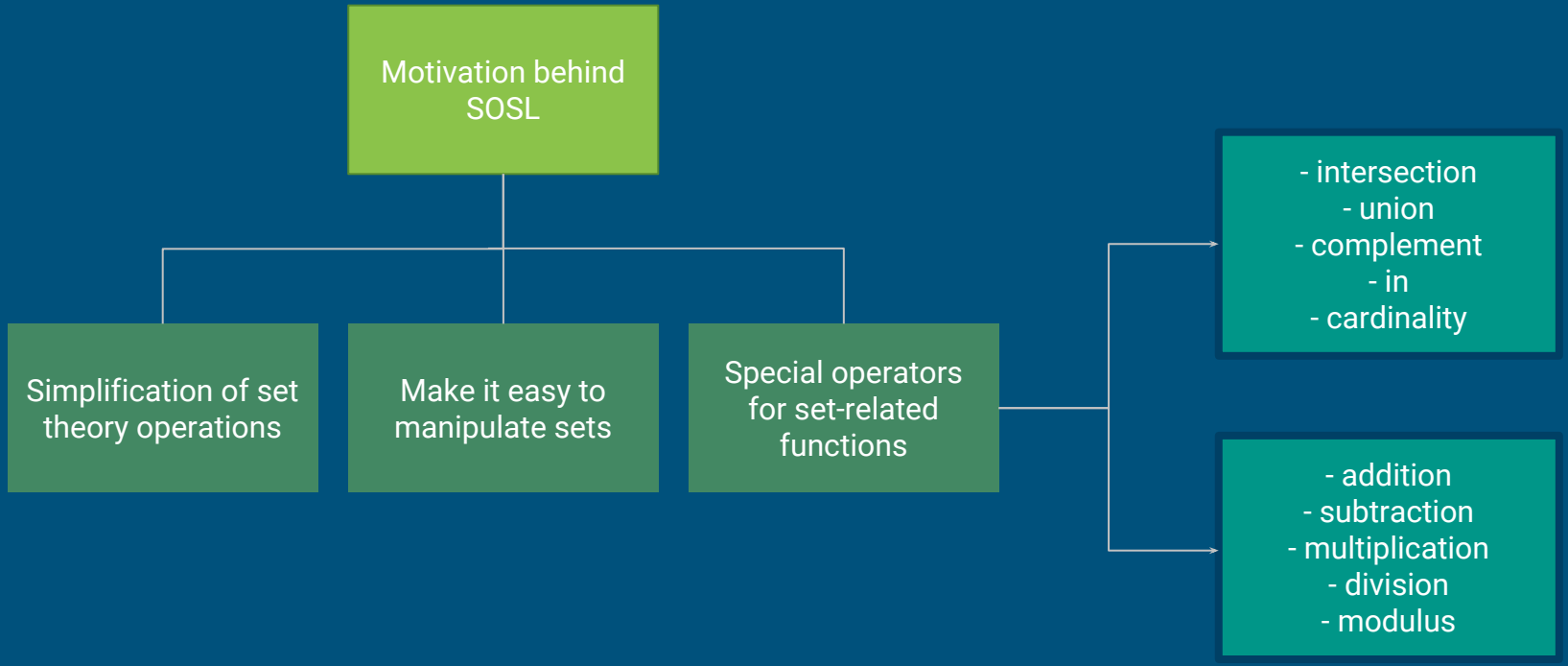
:{Set Operation Simplification Language}:

Introduction



Operators for set-related functions

		EXAMPLES
:u	UNION	• :{1,2}: :u :{1,3,4}: = :{1,2,3,4}: • :{:{1,4,5}:,6,:{7,8}:}: :u :{5}: = :{:{1,4,5}:,6,:{7,8}:,5}:
:n	INTERSECTION	• :{1,2}: :n :{1,3,4}: = :{1}: • :{:{1,4,5}:,6,:{7,8}:}: :n :{5}: = :{}: • :{:{1,2}:,5,6}: :n :{:{1,2}:,6,7}: = :{:{1,2}:,6}:
:i	IN	• :{1,3,4}: :i :{1,2}:is false • :{1,3,4}: :i 1 is true • 6 :i :{2,3}: returns an error
:c	COMPLEMENT	• :{1,2}: :c :{1,3,4}: = :{3,4}: • 1 :c :{1,3,4}: returns error • :{1,5,6}: :c :{1,2,3,4,5,6}: = :{2,3,4}:
| |	CARDINALITY	• A = :{1,2,3,4,5}:; |A| returns 5

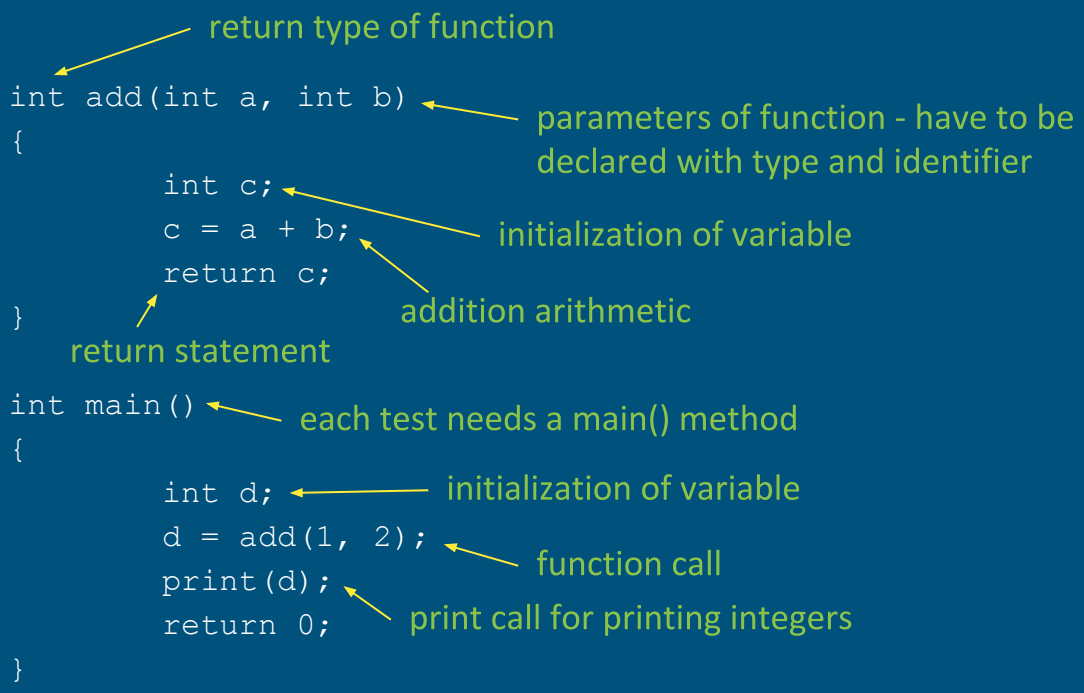
Order of Operations

- Control flow:
  - if, for, forEach
- Order of operations: Set operations are evaluated left to right in following the following hierarchy: (),:u = :n,:c,:i. :i has the lowest since the left and right sides of an :i expression must be completely evaluated before :i can. Since :u and :n have equal order, they will be evaluated left to right.

A :c C :u A :n B
is equivalent to A:c ((C :u A) :n B).

A :c C :n D :i B :u C
is equivalent to (A: c (C :n D)) :i (B :u C)

Syntax


return type of function
int add(int a, int b)
parameters of function - have to be declared with type and identifier
{
int c;
c = a + b;
initialization of variable
addition arithmetic
return c;
}
return statement
int main()
each test needs a main() method
{
int d;
initialization of variable
d = add(1, 2);
function call
print(d);
print call for printing integers
return 0;
}

Set-related functions

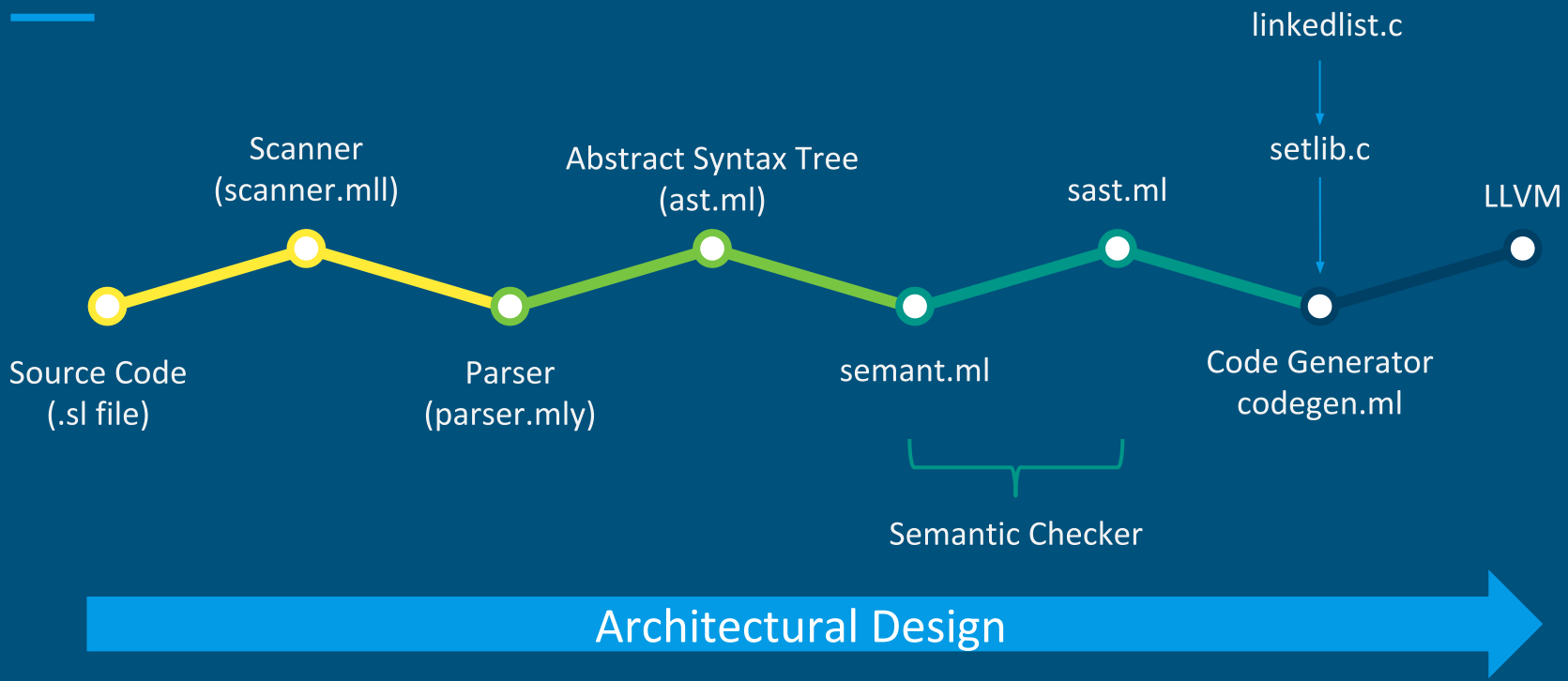
```
void create_set(int elmType)

void *adds(void *set_ptr, void *value)

void destroy(void *set_ptr)

**All other set related functions are tied to operators.
```

Overall Architecture



Testing & Debugging

- Shell script (testall.sh) for automated testing
- Include both fails and tests
  - Some standard tests from microc
  - More tests for our specific set functions
- Verified that test cases pass before committing when possible
- Debugging included adding print statements at different points of setlib.c to see why output from some tests was different from what was anticipated
  - Time consuming part of testing

Roadblocks & Lessons Learned

- Time Constraints
- Underestimating Scope of work
  - Starting Late
- Initial debug of parsing errors
- Too much reverse engineering
  - Read more documentation
- Determining how to implement set type
  - Set Literal
  - Connecting set struct to codegen
- Recursive Functions

Demonstration

```
int main()
{
    set:{int}: a;
    a = :{1,2,3}:;

    set:{int}: b;
    b = :{4,5,6}:;

    prints("OK");

    return 0;
}
```

- Variable Declaration
  - Nested set OK :{:{int}:}:
- SetLit assignment
- Set Operators
  - :u - Union : returns void *
  - :i - has : returns int (1,0)