Microteaching session

Oskar Abrahamsson

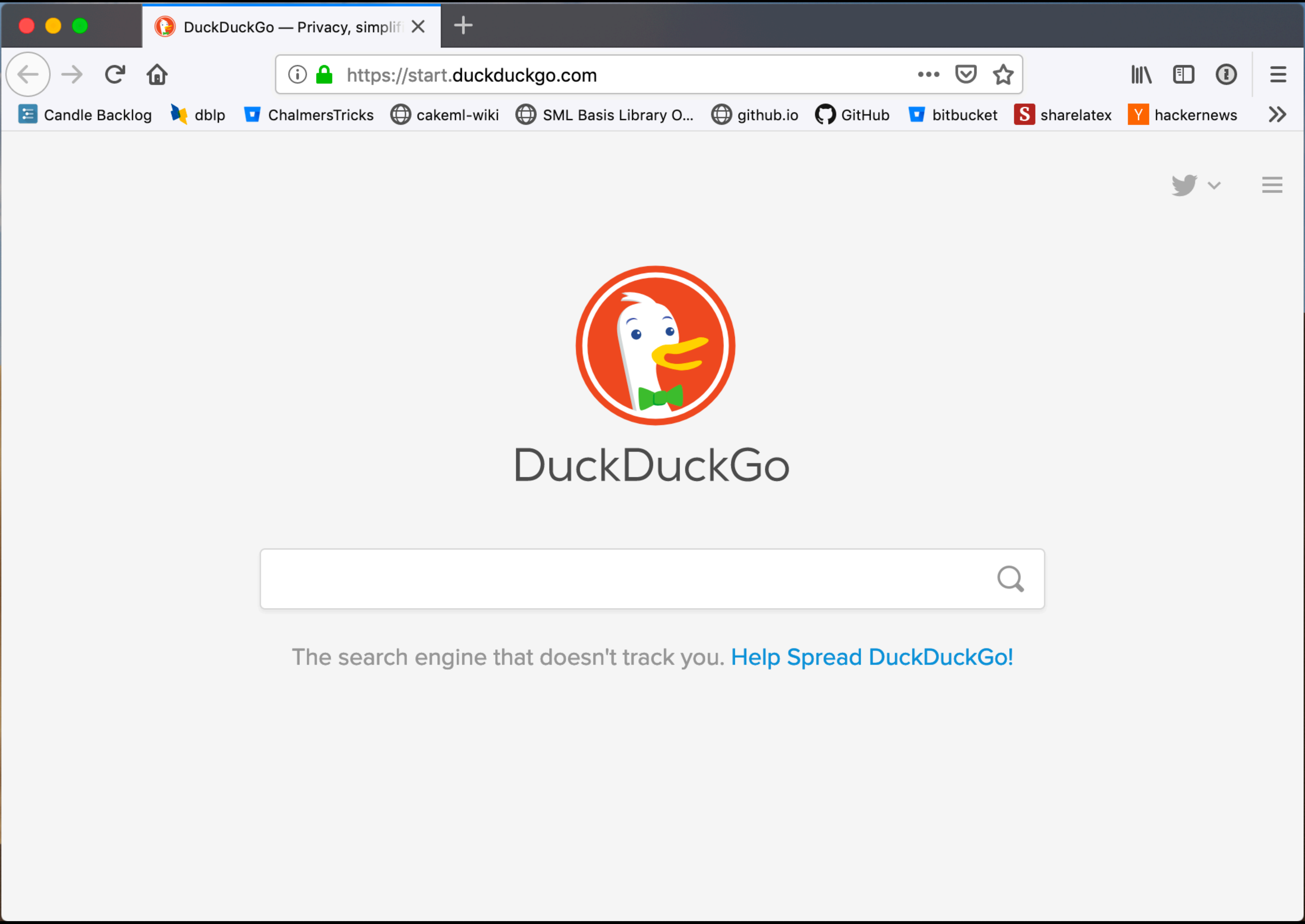
GFOK020, May 2019

Programming and compilers

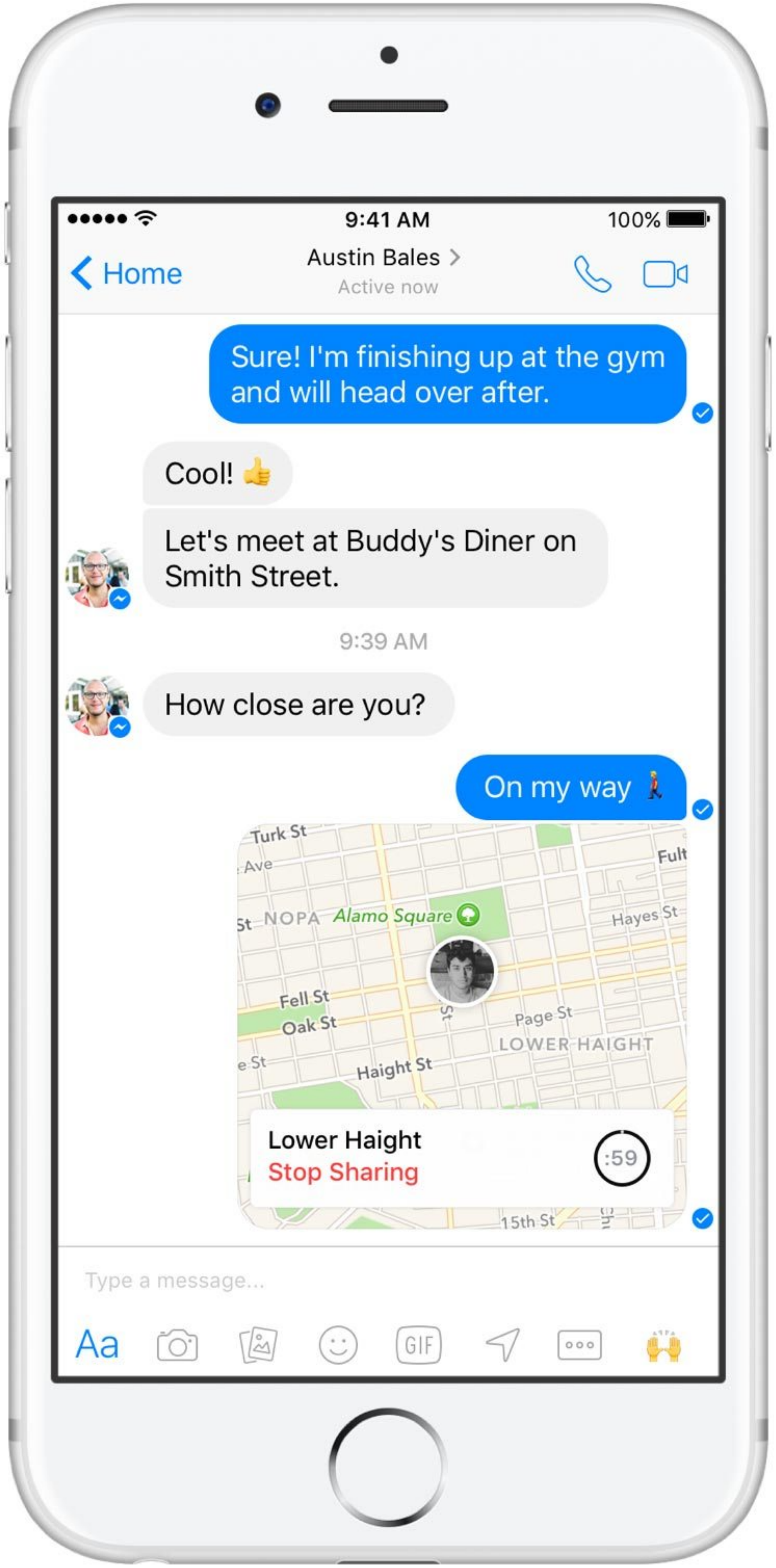
Goals

- Computer programs
- Programming languages
- Compilers

Computer programs



The search engine that doesn't track you. Help Spread DuckDuckGo!



Sure! I'm finishing up at the gym and will head over after.

Cool! 👍

Let's meet at Buddy's Diner on Smith Street.

9:39 AM

How close are you?

On my way 🚶

MARIO
× 4
★ × 18
TIME
245
× 84
106920

Programming
(a computer)

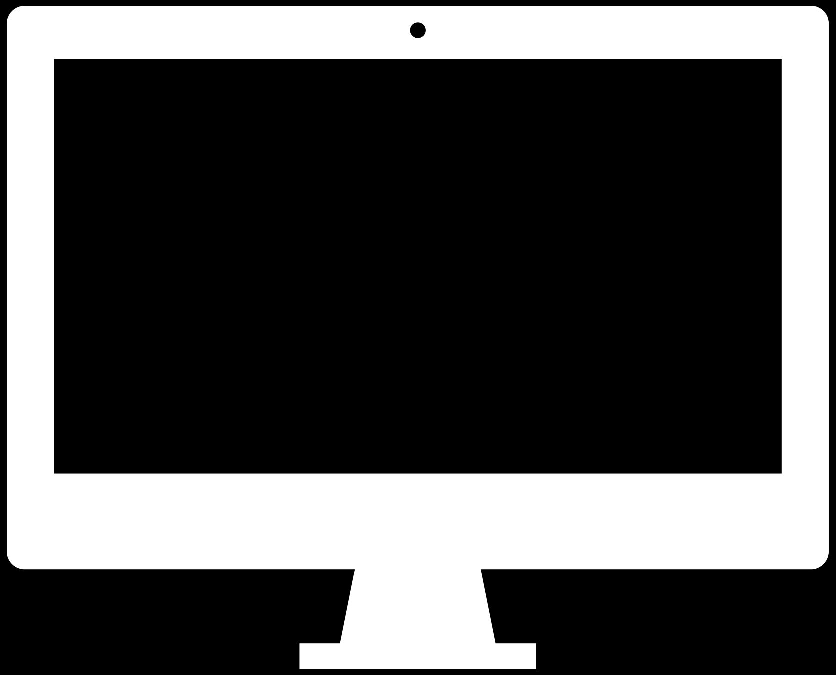
Example: Fibonacci sequence

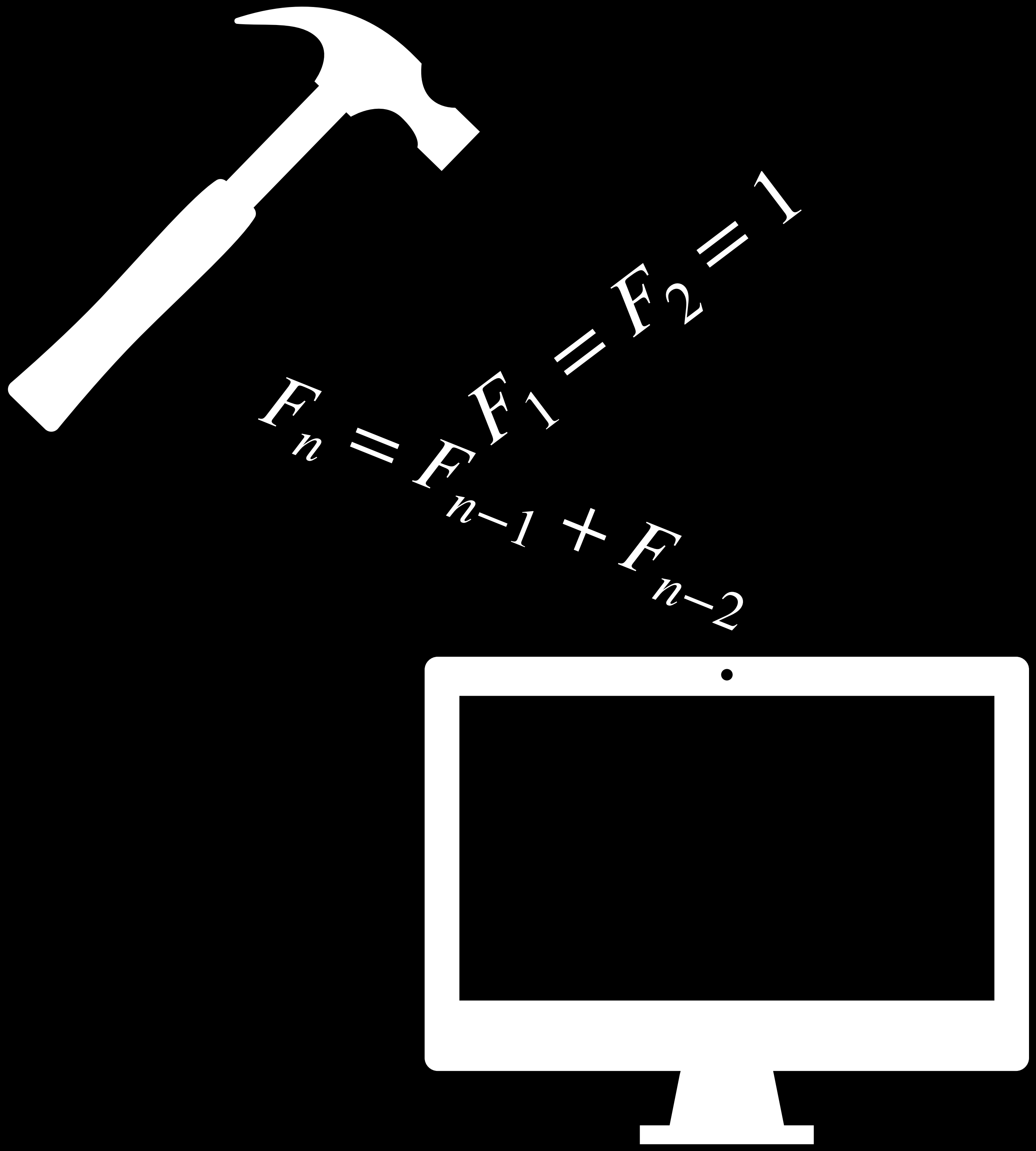
$$F_1 = F_2 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

$$F_n = F_{n-1} + F_{n-2}$$

$$F_1 = F_2 = 1$$



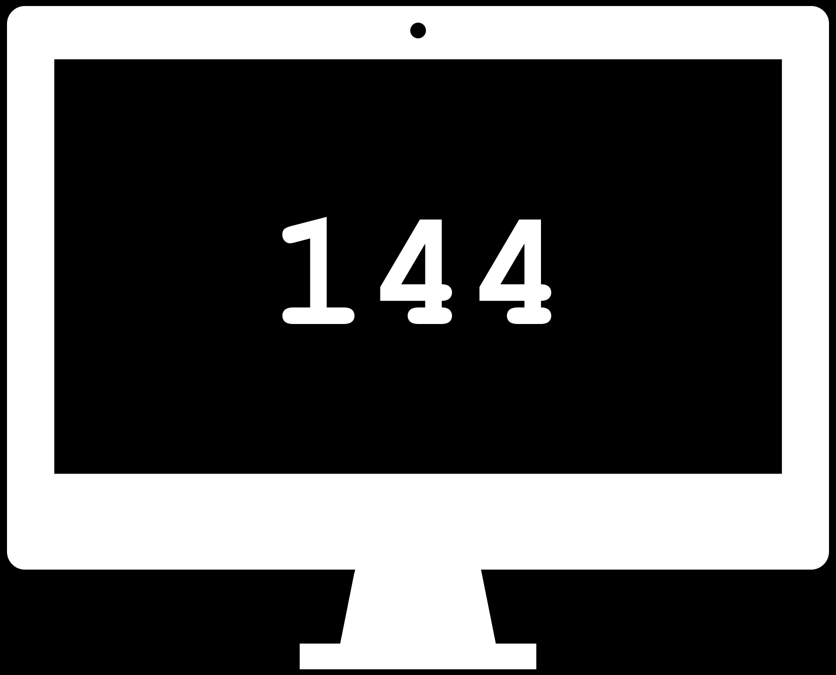


$$F_1 = F_2 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

$F_1 = F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$



144

Example: Fibonacci sequence

$$F_1 = F_2 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

Example: Fibonacci sequence

$$F_1 = F_2 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

$$F_{n-1} = F_n - F_{n-2}$$

Example: Fibonacci sequence

```
41 bb 0a 00 00 00 41 bc 01 00 00 00
41 bd 01 00 00 00 4d 01 ec 4d 89 ee
4d 89 e5 4d 29 f5 49 ff cb 75 ef 4c
89 e6 48 8d 3d 00 00 00 00 b0 00 51
e8 00 00 00 00 59 c3
```

```
_main:
  mov   r11, 10
  mov   r12, 1
  mov   r13, 1

.loop:
  add   r12, r13
  mov   r14, r13
  mov   r13, r12
  sub   r13, r14
  dec   r11
  jne   .loop

  mov   rsi, r12
  lea   rdi, [rel .fmt]
  mov   al, 0
  push  rcx
  call  _printf
  pop   rcx
  ret
```

```
_main:
  mov   r11, 10
  mov   r12, 1
  mov   r13, 1

.loop:
  add   r12, r13
  mov   r14, r13
  mov   r13, r12
  sub   r13, r14
  dec   r11
  jne   .loop

  mov   rsi, r12
  lea   rdi, [rel .fmt]
  mov   al, 0
  push  rcx
  call  _printf
  pop   rcx
  ret
```

$$F_n = F_{n-1} + F_{n-2}$$

```
_main:
  mov    r11, 10
  mov    r12, 1
  mov    r13, 1

 .loop:
  add    r12, r13
  mov    r14, r13
  mov    r13, r12
  sub    r13, r14
  dec    r11
  jne    .loop

  mov    rsi, r12
  lea    rdi, [rel .fmt]
  mov    al, 0
  push   rcx
  call   _printf
  pop    rcx
  ret
```

$$F_n = F_{n-1} + F_{n-2}$$

$$F_{n-1} = F_n - F_{n-2}$$

```
_main:
  mov   r11, 10
  mov   r12, 1
  mov   r13, 1

.loop:
  add   r12, r13
  mov   r14, r13
  mov   r13, r12
  sub   r13, r14
  dec   r11
  jne   .loop

  mov   rsi, r12
  lea   rdi, [rel .fmt]
  mov   al, 0
  push  rcx
  call  _printf
  pop   rcx
  ret
```

$$F_n = F_{n-1} + F_{n-2}$$

$$F_{n-1} = F_n - F_{n-2}$$

?

```
int main() {
  int a = 1;
  int b = 1;

  for (int n = 0; n < 10; n++) {
    a = a + b;
    b = a - b;
  }

  printf("%d\n", a);

  return 0;
}
```

```c
int main() {
  int a = 1;
  int b = 1;
```

For all n in the range 0 to 9;

```c
  for (int n = 0; n < 10; n++) {
    a = a + b;
    b = a - b;
  }

  printf("%d\n", a);

  return 0;
}
```

```
int main() {
  int a = 1;
  int b = 1;

  for (int n = 0; n < 10; n++) {
    a = a + b;
    b = a - b;
  }

  printf("%d\n", a);

  return 0;
}
```

... do this,

```
int main() {
  int a = 1;
  int b = 1;

  for (int n = 0; n < 10; n++) {
    a = a + b;
    b = a - b;
  }

  printf("%d\n", a);

  return 0;
}
```

… and then print something.

```
def fib(n):
    a = 1
    b = 1
    for i in range(0, n):
        a = a + b
        b = a - b
    print(a)
```

```
fun fib n =
  if n < 2 then 1 else
    fib (n - 1) + fib (n - 2);
```

```
int main() {
  int a = 1;
  int b = 1;

  for (int n = 0; n < 10; n++) {
    a = a + b;
    b = a - b;
  }

  printf("%d\n", a);

  return 0;
}
```

```
41 bb 0a 00 00 00 41 bc 01 00 00 00

41 bd 01 00 00 00 4d 01 ec 4d 89 ee

4d 89 e5 4d 29 f5 49 ff cb 75 ef 4c

89 e6 48 8d 3d 00 00 00 00 b0 00 51

e8 00 00 00 00 59 c3
```

Compilers

A compiler is …

A compiler is …

- A computer program

A compiler is …

- A computer program
- A translator from language A to language B

A compiler is …

Typically a high-level programming language

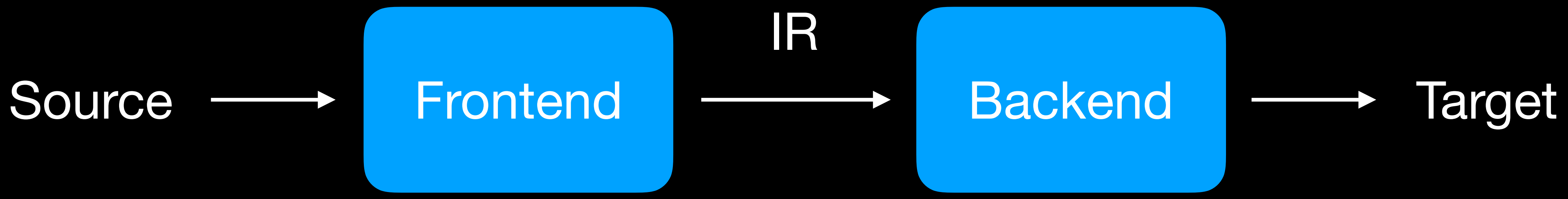
- A computer program
- A translator from language A to language B

A compiler is …

- A computer program
- A translator from language A to language B

Typically a low-level thing, e.g. machine code

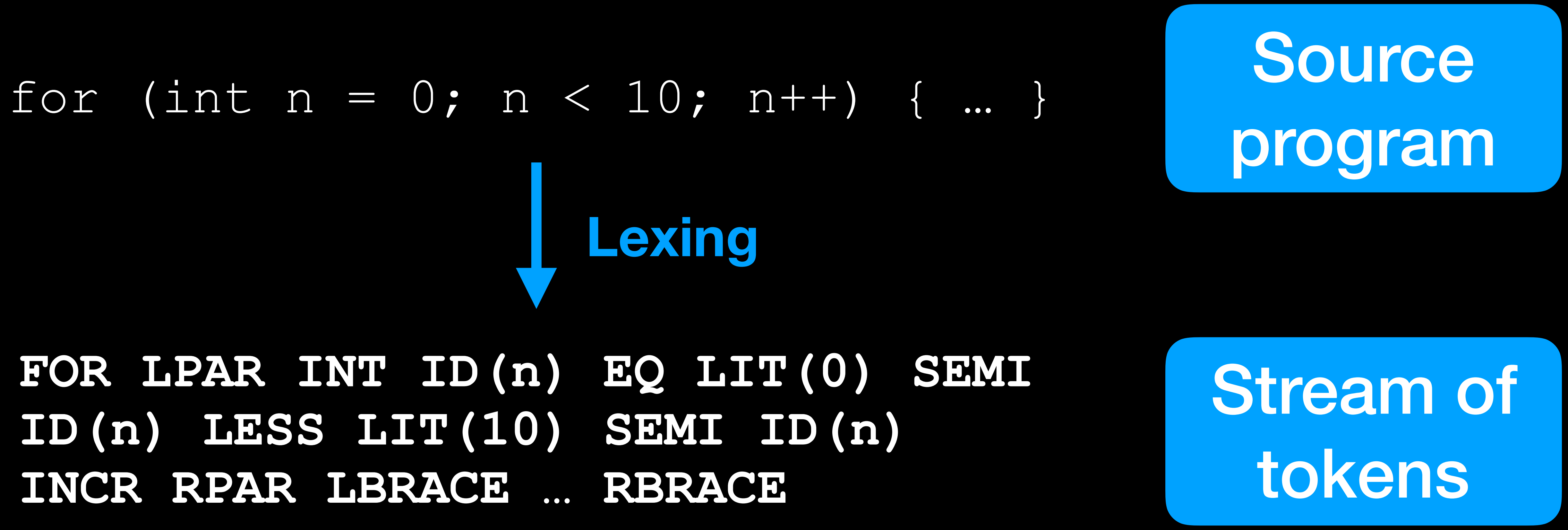
Basic structure



- Frontend:
  - Lexing
  - Parsing
  - (Type-checking)

```
for (int n = 0; n < 10; n++) { … }
```

Source program



```
for (int n = 0; n < 10; n++) { … }
```

Source program

Lexing

```
FOR LPAR INT ID(n) EQ LIT(0) SEMI
ID(n) LESS LIT(10) SEMI ID(n)
INCR RPAR LBRACE … RBRACE
```

Stream of tokens

```
for (int n = 0; n < 10; n++) { … }
```

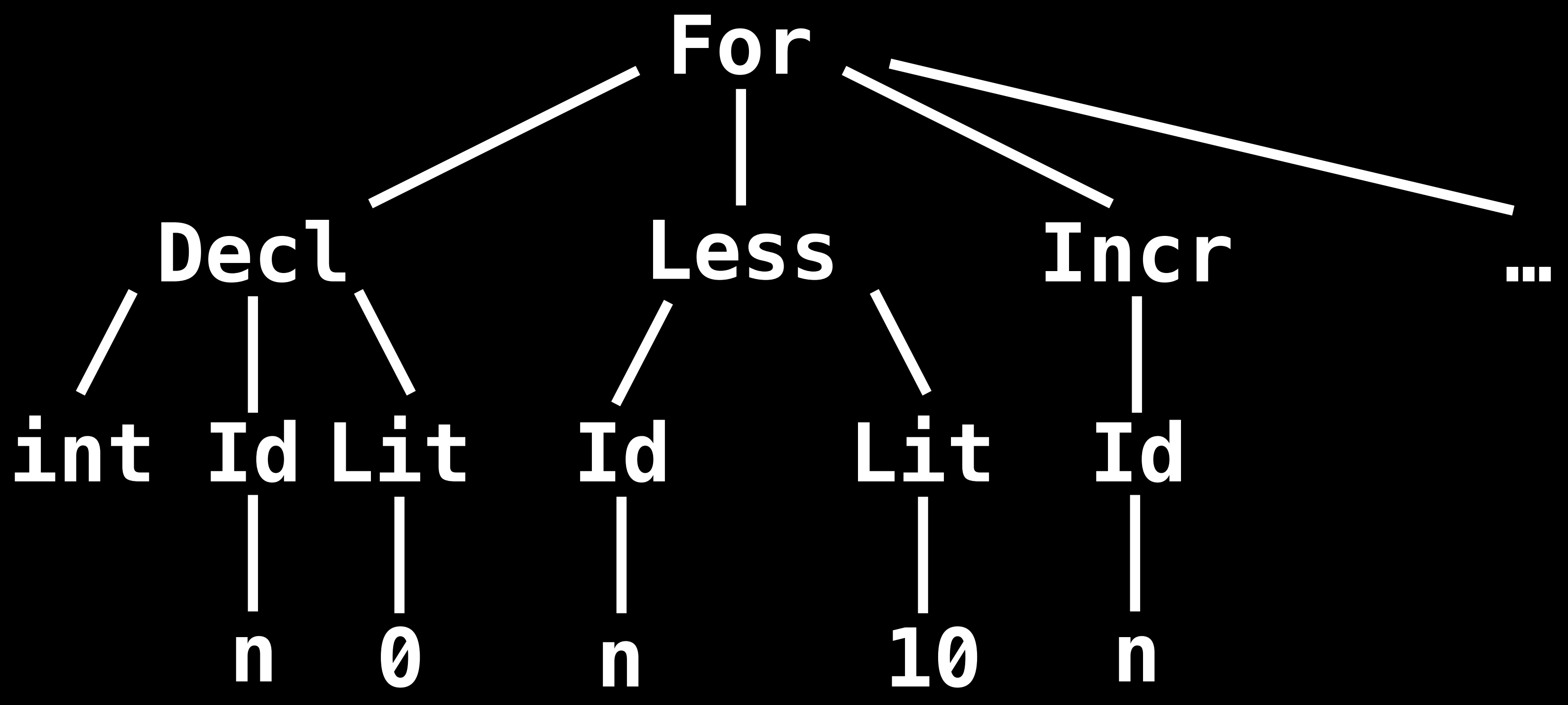
Source program

Lexing

```
FOR LPAR INT ID(n) EQ LIT(0) SEMI
ID(n) LESS LIT(10) SEMI ID(n)
INCR RPAR LBRACE … RBRACE
```

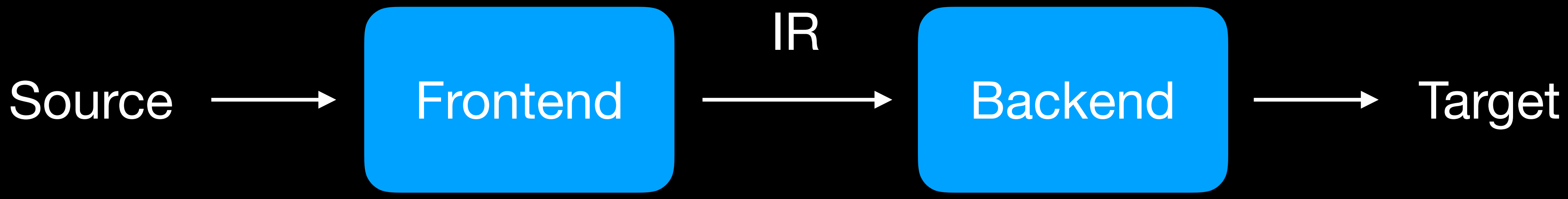
Stream of tokens

Parsing

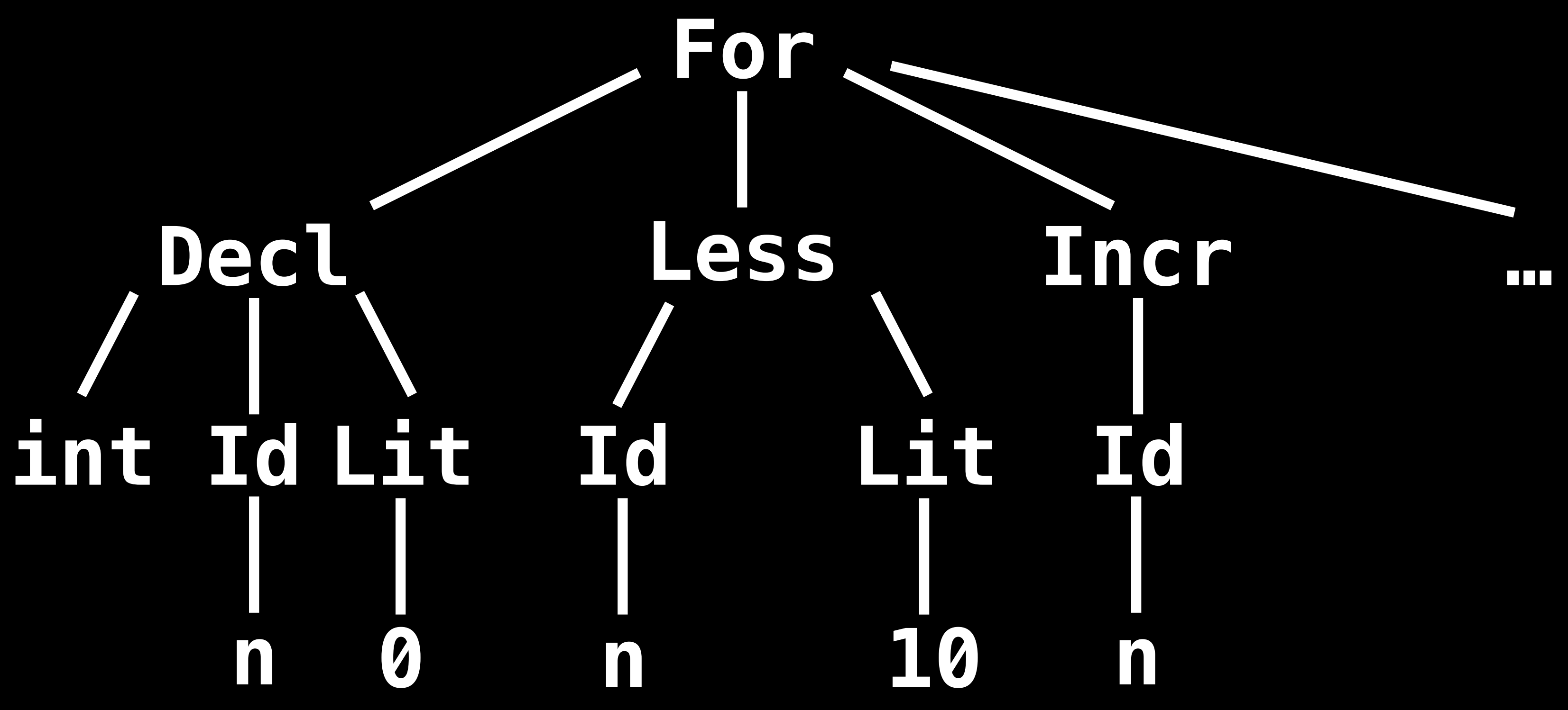


Abstract syntax

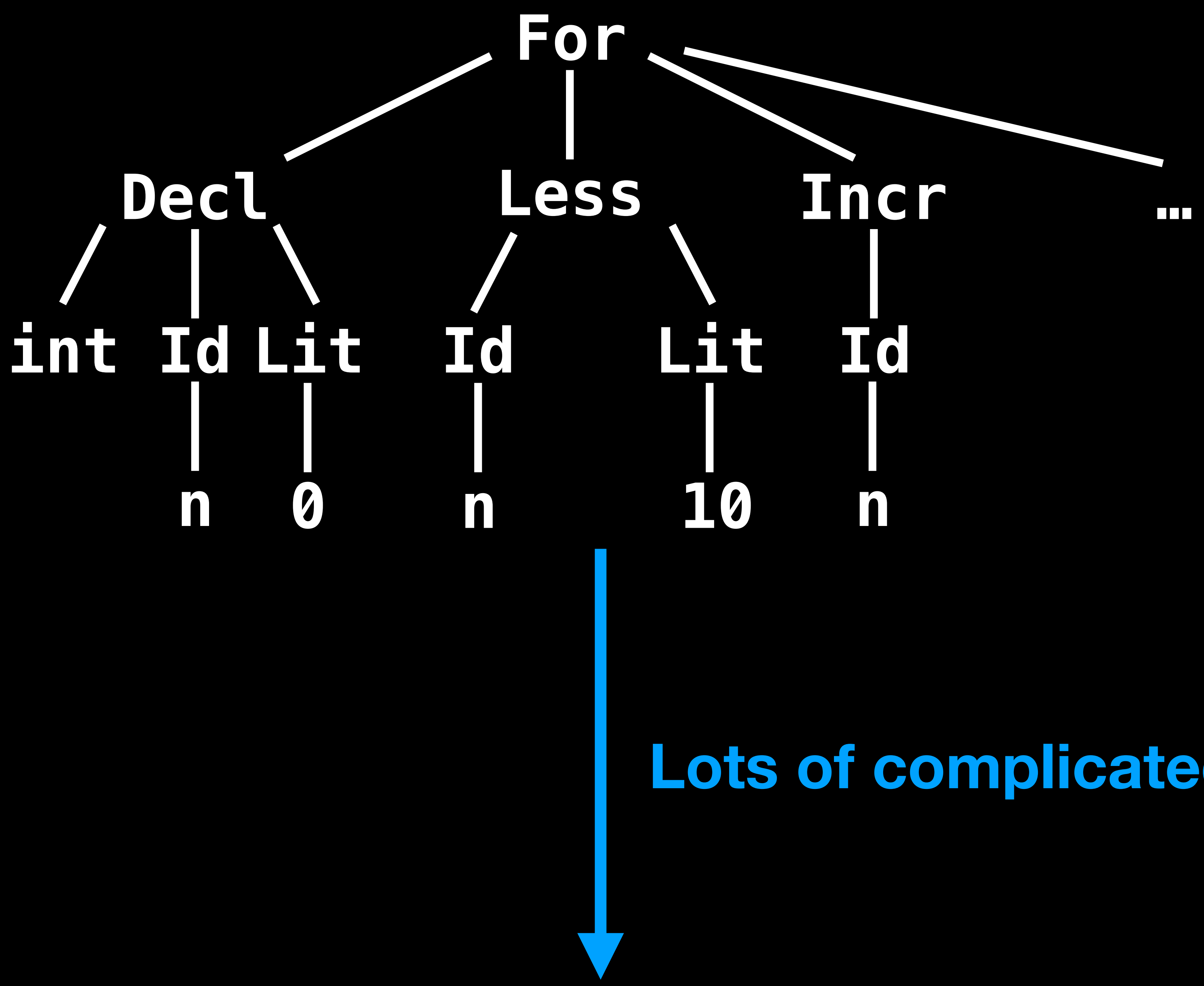
Basic structure



- Backend:
  - Analysis
  - Optimization
  - Emit code in target language



Abstract syntax



Abstract syntax

Lots of complicated stuff

```
41 bb 0a 00 00 00 41 bc 01 00 00 00
41 bd 01 00 00 00 4d 01 ec 4d 89 ee
4d 89 e5 4d 29 f5 49 ff cb 75 ef 4c
89 e6 48 8d 3d 00 00 00 00 b0 00 51
e8 00 00 00 00 59 c3
```

Target

Compiling is difficult

- Source language structured into:
    - functions
    - statements
    - expressions
    - etc

Compiling is difficult

- Machine code structured into:
  - sequences of instructions
  - (conditional) jumps
  - memory manipulation
  - etc

- Optimizations (time)
- Optimizations (space, memory)
- Multiple targets
- Interface with IDEs (e.g. Visual Studio)

Some questions

- What is a compiler?

- A compiler is a computer program that makes computer programs. How are they made?

  Are they compiled?

- Some compilers are written in the language they are compiling.

  How would you compile your compiler written in language X, if your compiler is the first compiler that compiles language X?