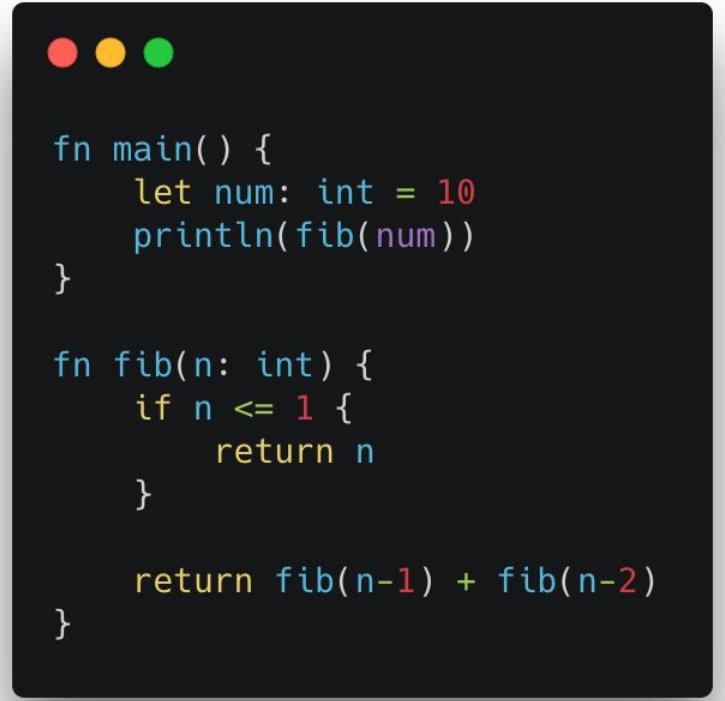

a p p
m o t
i o n

From A to B;
Compiler Internals

https://github.com/garritfra/sabre

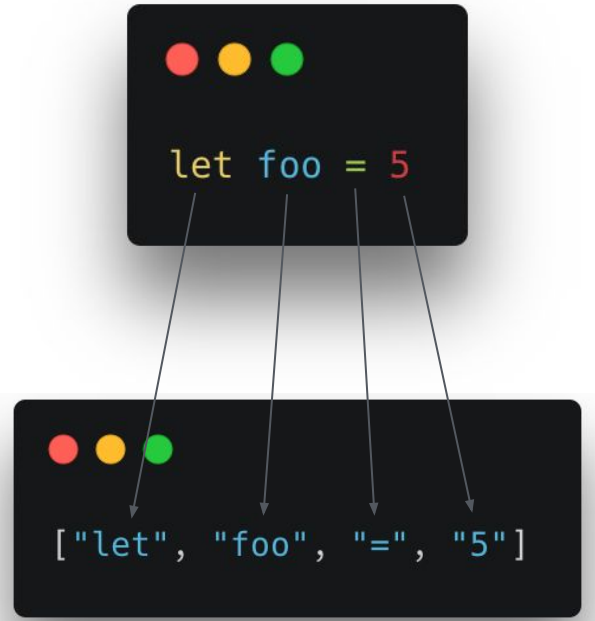
A bullshit-free (©) programming language



```
fn main() {
    let num: int = 10
    println(fib(num))
}

fn fib(n: int) {
    if n <= 1 {
        return n
    }

    return fib(n-1) + fib(n-2)
}
```

```
#include "stdio.h"

int main() {
    int num = 10;
    printf("%d\n", fib(num));
}

int fib(int n) {
    if (n >= 1) {
        return n;
    }

    return fib(n-1) + fib(n-2);
}
```

Tokenizing: Process

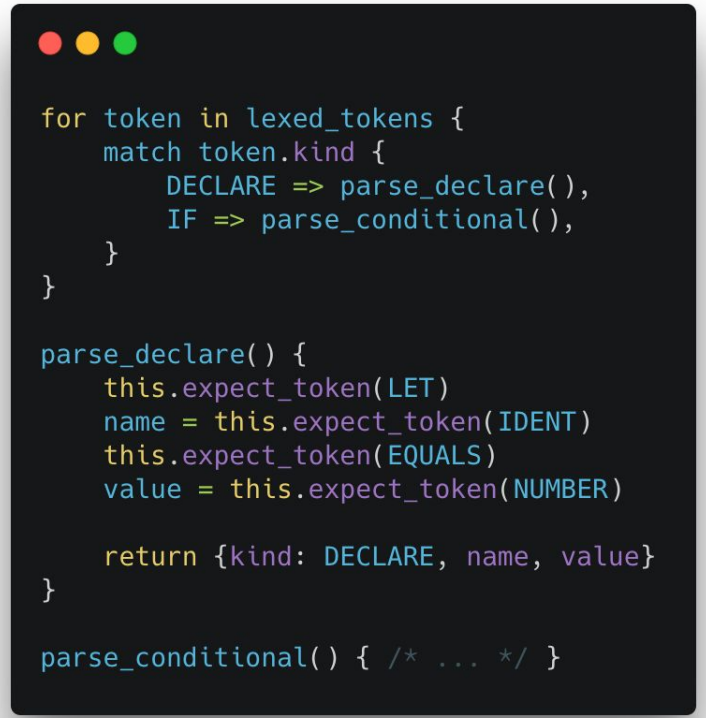


Lexing: Process

Lexing: Algorithm

Parsing: Process

Parsing: Algorithm

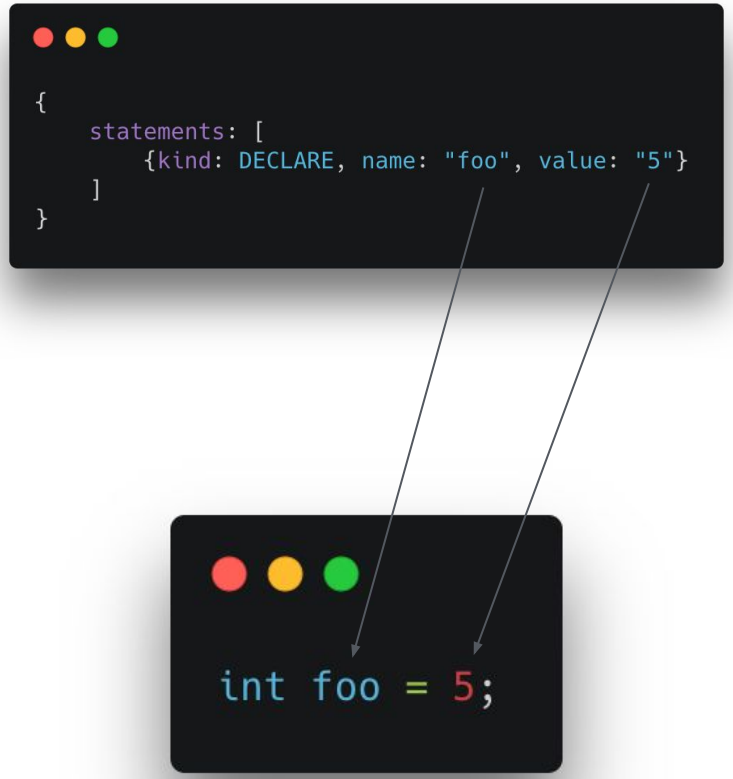


```
for token in lexed_tokens {
    match token.kind {
        DECLARE => parse_declare(),
        IF => parse_conditional(),
    }
}

parse_declare() {
    this.expect_token(LET)
    name = this.expect_token(IDENT)
    this.expect_token(EQUALS)
    value = this.expect_token(NUMBER)

    return {kind: DECLARE, name, value}
}

parse_conditional() { /* ... */ }
```

Parsing: Real life Example

```
fn main() {
    println(greet("World"))
}

fn greet(name: string): string {
    return "Hello " + name
}
```

Generating: Process



Generating: Algorithm

```
for statement in statements {
    match statement.kind {
        DECLARE => generate_declare(statement),
        ... => ...
    }
}

generate_declare(statement) {
    return "int ${statement.name} = ${statement.value};"
}
```

Result

```
fn main() {
    let num: int = 10
    println(fib(num))
}

fn fib(n: int) {
    if n <= 1 {
        return n
    }

    return fib(n-1) + fib(n-2)
}
```

```
#include "stdio.h"

int main() {
    int num = 10;
    printf("%d\n", fib(num));
}

int fib(int n) {
    if (n >= 1) {
        return n;
    }

    return fib(n-1) + fib(n-2);
}
```

Links & Resources

- https://github.com/garritfra/sabre
- http://www.buildyourownlisp.com/contents
- https://compilerbook.com/

Vielen Dank.

appmotion GmbH
Kleine Freiheit 68
22767 Hamburg

040 - 228 200 600
kontakt@appmotion.de