I-Lang

PT:
Евгений Букреев
Владислав Феофилактов
Даниил Бакин

Описание проекта

- Императивный язык I
- **ANTLR/LLVM/Kotlin**
- Встроенные типы – **integer**, **real**, **boolean**
- Массивы и записи, рутины
- Циклы **while** и **for**
- Конструкции **if**, **return**, **break**, **continue**
- **+ | - | * | / | % | < | <= | > | >= | = | /= | and | or | xor**

Команда - PT

Владислав Феофилактов -

лексер и парсер, кодген

Евгений Букреев -

IR и его построение, лаунчер

Даниил Бакин -

семантические проверки, тайпчек

Вычисление чисел Фибоначчи

```
routine fib(n: integer) : integer is
    if n = 1 then
        return 1
    end

    var num1 : integer
    num1 := 0

    var num2 is 1
    var nextNum : integer is num1

    for i in 1..(n - 1) loop
        nextNum := num1 + num2
        num1 := num2; num2 := nextNum
    end

    return nextNum
end
```

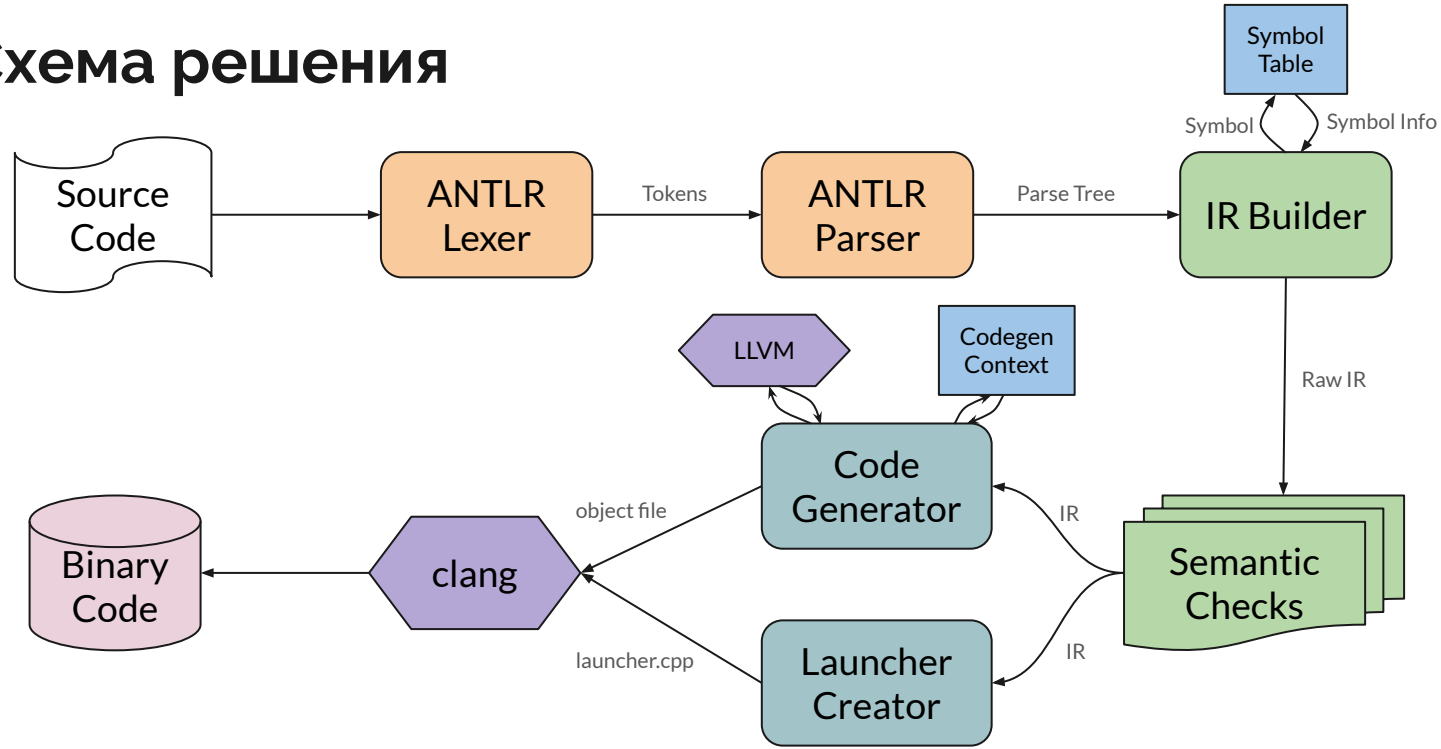
Полный двоичный сумматор

```
type AdderResult is record
    var sum : boolean
    var c_out : boolean
end

routine fullAdder(a : boolean, b : boolean, c_in: boolean) : AdderResult is
    var sum is c_in xor a xor b
    var c_out is (a and b) or (b and c_in) or (a and c_in)

    var result: AdderResult
    result.sum := sum
    result.c_out := c_out

    return result
end
```

Сортировка пузырьком

```
routine bubbleSort(arr : array[] integer) is
    var current is arr[1]
    var temp : integer
    var i : integer is 1

    while i < (arr.size + 1) loop
        var j is 2
        while j <= (arr.size - i + 1) loop
            if arr[j-1] > arr[j] then
                temp := arr[j-1]
                arr[j-1] := arr[j]
                arr[j] := temp
            end
            j := j + 1
        end
        i := i + 1
    end
end
```

Схема решения

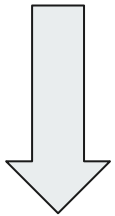


Лексер

```
lexer grammar iLangLexer;

...
VAR : 'var';
IS : 'is';
TYPE : 'type';
ROUTINE : 'routine';
RETURN : 'return';
BREAK : 'break';
CONTINUE : 'continue';
END : 'end';
COLON : ':';
SEMICOLON : ';';
...
```

Лексер

```
routine fib(n: integer) : integer is
```



```
[@0,0:6='routine',<4>,1:0]
[@2,8:10='fib',<48>,1:8]
[@3,11:11='(',<11>,1:11]
[@4,12:12='n',<48>,1:12]
[@5,13:13=':',<9>,1:13]
[@7,15:21='integer',<42>,1:15]
[@8,22:22=')',<12>,1:22]
[@10,24:24=':',<9>,1:24]
[@12,26:32='integer',<42>,1:26]
[@14,34:35='is',<2>,1:34]
```

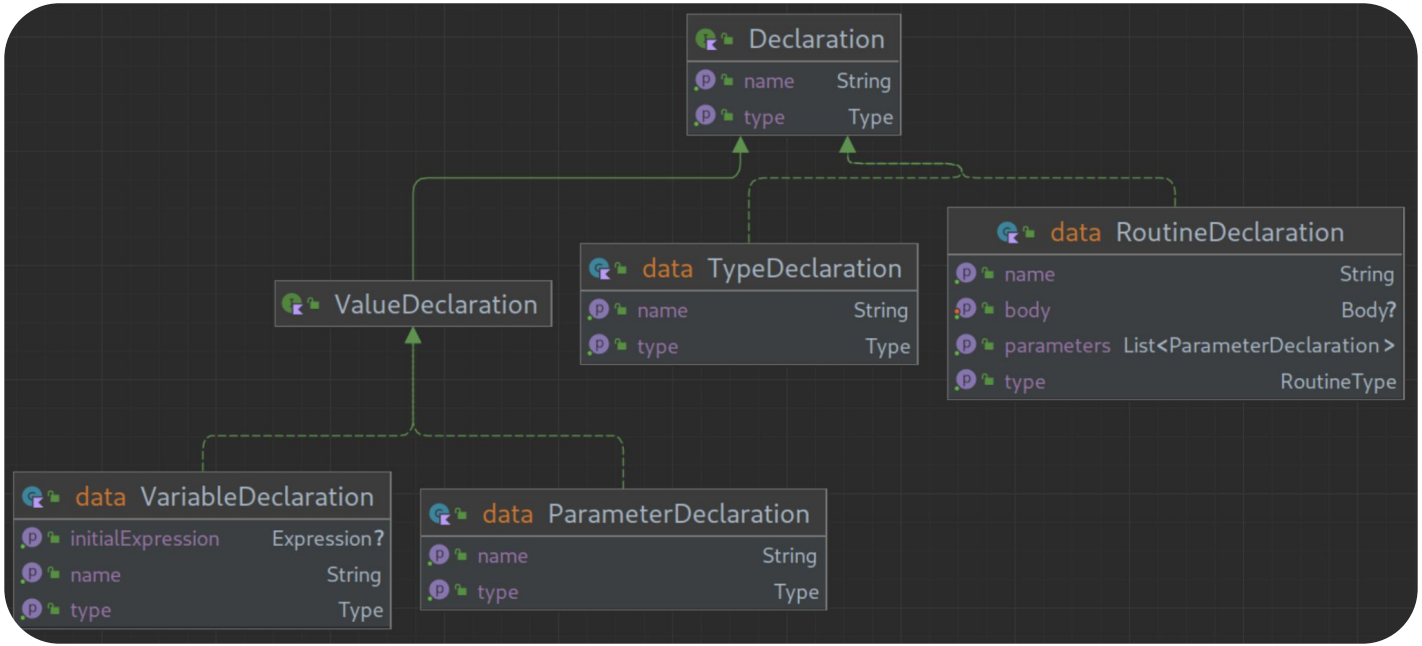
Парсер

```
parser grammar iLangParser;

...
simpleDeclaration
    : variableDeclaration
    | typeDeclaration
    ;

variableDeclaration
    : VAR Identifier COLON type (IS expression)?
    | VAR Identifier IS expression
    ;

routineDeclaration
   : ROUTINE Identifier (L_PARENTHESIS parameters? R_PARENTHESIS)? (COLON type)? IS
     body
     END
...
```

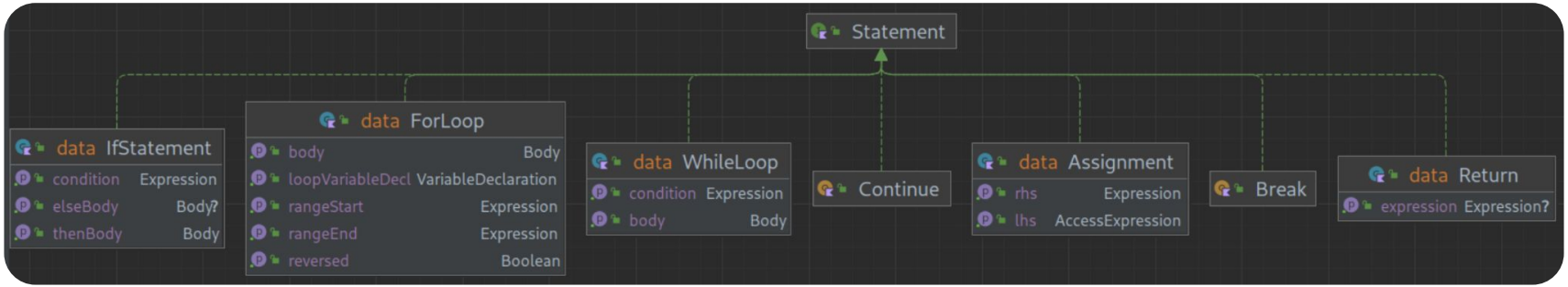
Парсер

```
(program
    (routineDeclaration routine fib
        (
            (parameters
                (parameterDeclaration n :
                    (type
                        (primitiveType integer)))) ) :
        (type
            (primitiveType integer)) is
        (body
            (statement
                (ifStatement if
                    (expression
                        (expression
                            (primary
...
```

Промежуточное представление



Промежуточное представление



Промежуточное представление

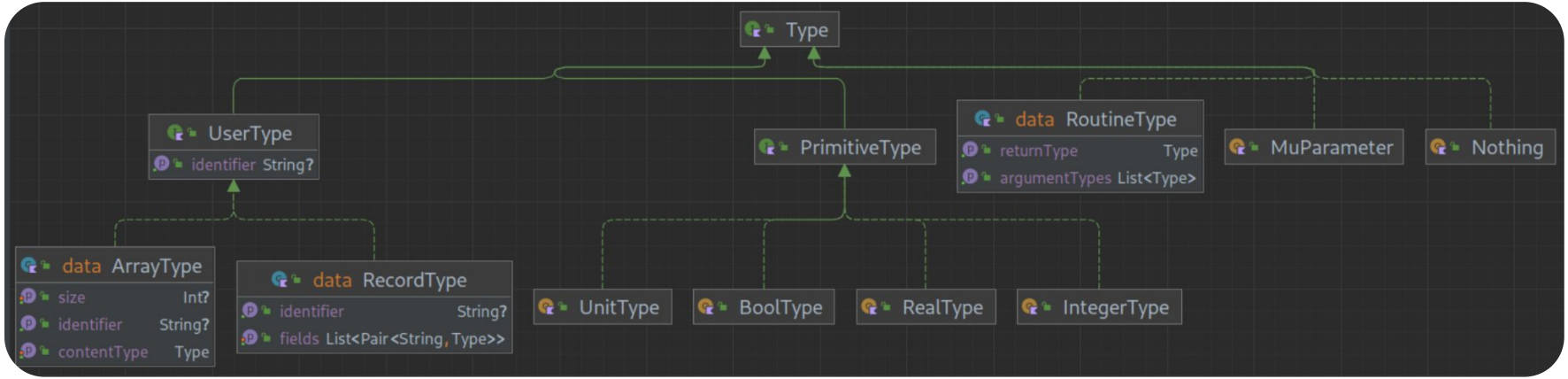
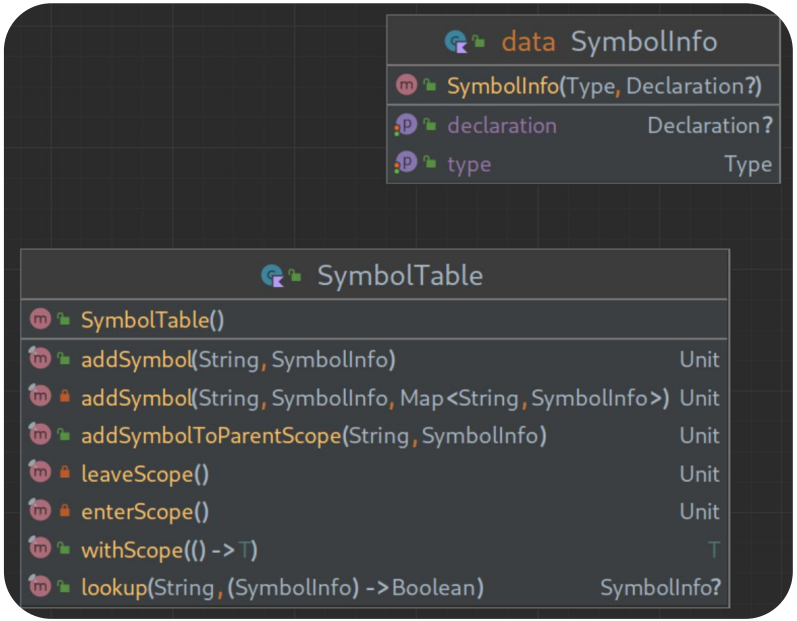


Таблица символов



Внутреннее представление

```
Program(
    declarations=[
        RoutineDeclaration(
            name=$fib,
            type=RoutineType(
                argumentTypes=[
                    IntegerType
                ],
                returnType=IntegerType
            ),
            parameters=[
                ParameterDeclaration(
                    name=n,
                    type=IntegerType
                )
            ],
...
```

Семантические проверки

```
class SemanticStageProcessor {
    private val analysers = listOf(
        FunctionReturnAnalyzer()
    )

    private val checkers = listOf(
        TypeChecker(),
        ArraySizeModificationIsProhibited(),
        IdentifierChecker(),
        FunctionReturnChecker(),
        BreakAndContinueInsideCyclesChecker(),
        AssignNewValueToArgument(),
        ForRangeIsInteger(),
    )

    private val transformers = listOf(
        DeadCodeEliminator(),
    )
}
```

Кодогенерация

```
    fun generate(program: Program) {
        initializeLlvm()

        for (declaration in program.declarations) {
            processTopLevelDeclaration(declaration)
        }

        for (declaration in program.declarations) {
            processDeclaration(declaration)
        }

        LLVMDumpModule(module)

        val verificationResult = LLVMVerifyModule(
                                 module, LLVMPrintMessageAction, errorBuffer)
        if (verificationResult != 0) {
            report("function verification error!\n${errorBuffer.string}")
        }
}
```

Кодогенерация

```
define i32 @"$fib"(i32 %0) {
entry:
  %binary-op-ints = icmp eq i32 %0, 1
  %condition = icmp eq i1 %binary-op-ints, true
  br i1 %condition, label %if-then, label %if-else
if-then:                                          ; preds = %entry
  ret i32 1
if-else:                                          ; preds = %entry
  br label %if-merge
if-merge:                                         ; preds = %if-else
  %num1 = alloca i32, align 4
  store i32 0, ptr %num1, align 4
  store i32 0, ptr %num1, align 4
  %num2 = alloca i32, align 4
  store i32 1, ptr %num2, align 4
```

Лаунчер

```
extern "C" {
    int $fib(int);
}

int main(const int argc, const char *argv[]) {
    if (strcmp(name, "$fib") == 0) {
        if (argc != 3) {
            // вывод ошибки
            return 1;
        }

        std::cout << $fib(atoi(argv[2])) << std::endl;
        return 0;
    }

    // вывод ошибки
    return 1;
}
```

Тестовая инфраструктура

- **Много тестовых данных** (28 файлов с исходниками, 49 рутин, 862 строки)
- **Виды тестов:**
  1. Тесты **семантических проверок** (36 тестов)
  2. Тесты каждого из **этапов компиляции** (всего 108 теста)

  Тесты парсера и лексера, построения IR, кодогенерации, E2E

Тестовая инфраструктура

- Небольшой **подъязык для запуска тестов кодогенерации** и **E2E** тестов компилятора:

```
// fact(1): 1
// fact(2): 2
// fact(3): 6
// fact(4): 24

routine fact(n: integer) : integer is
    if n >= 1 then
        return n * fact(n - 1)
    else
        return 1
    end
end
```

Что сделали дополнительно

1. Операторы контроля управления: **return**, **break**, **continue**
2. Определение длины массива: array**.size**
3. Проверка поля записи на инициализованность:
   record.field **= uninitialized**
   record.field **/= uninitialized**

Демо

Исходные коды / примеры