Поток выполнения

Лекция №9

Управление потоком выполнения

Структурное программирование:

1. Отказ от goto
2. Ветвления
3. Циклы

Управление потоком выполнения

Структурное программирование:

1. Отказ от goto
2. Ветвления
3. Циклы

Процедурное программирование:

1. Функции (процедуры)
2. Структуры (записи)

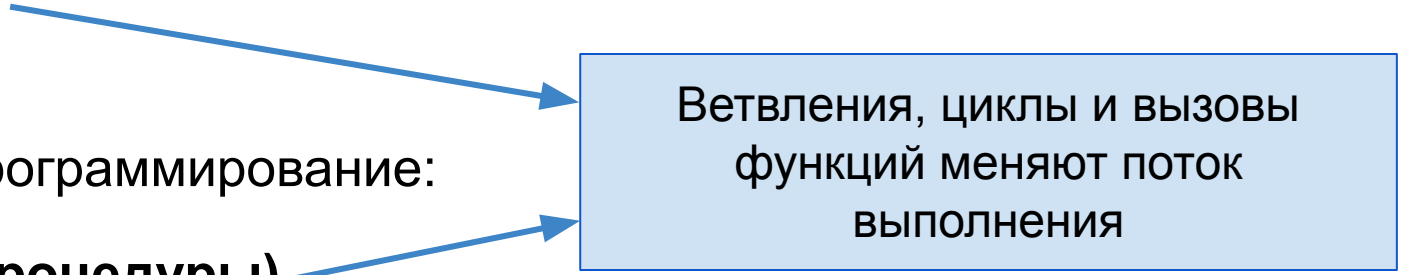
Управление потоком выполнения

Структурное программирование:

1. Отказ от goto
2. **Ветвления**
3. **Циклы**

Процедурное программирование:

1. **Функции (процедуры)**
2. Структуры (записи)

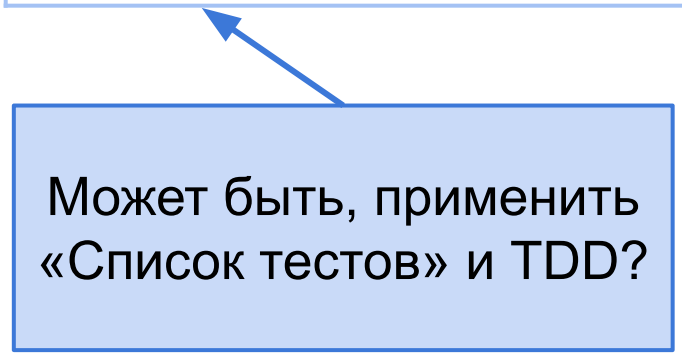


Проблема

Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	???

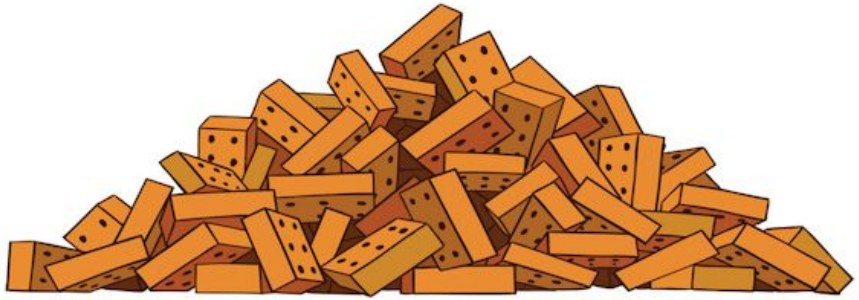
Проблема

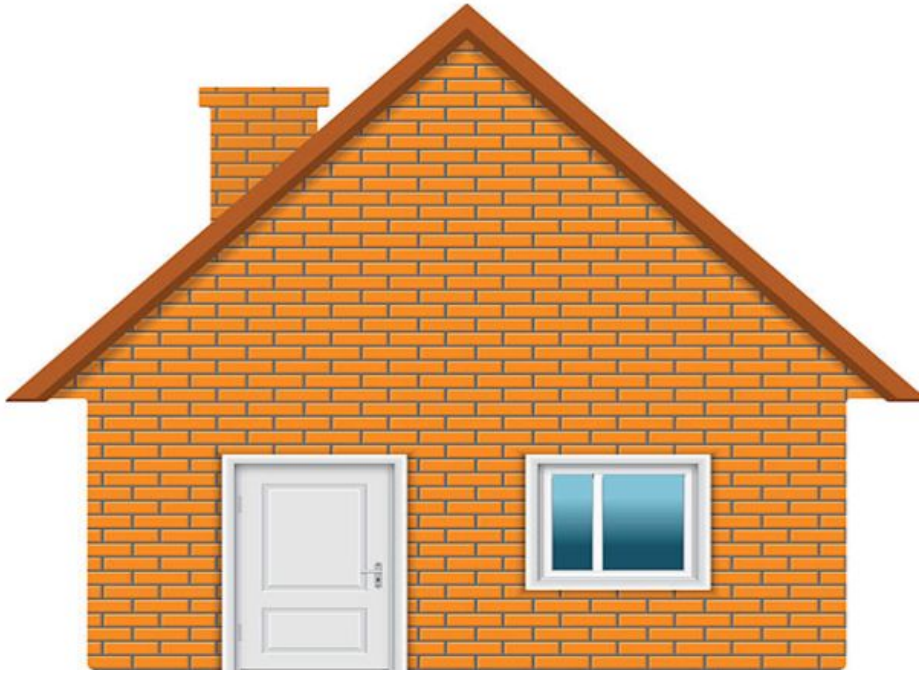
Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	???



Проблема

Модульных тестов недостаточно,
чтобы проверить **всю программу**





Истории в Agile

Проблема

Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	???

Проблема

Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	Написать 1. Пользовательскую историю 2. Критерии приёмки

Шаблон пользовательской истории (User Story)

Как {**роль**}, я хочу {**функция программы**}, чтобы {**решаемая задача**}

История №1

Как разработчик, я хочу иметь ветвления и рекурсию, чтобы ???

Шаблон:

Как {**роль**}, я хочу {**функция программы**}, чтобы {**решаемая задача**}

История №1

Как разработчик, я хочу иметь ветвления и рекурсию, чтобы считать числа Фибоначчи

Шаблон:

Как {**роль**}, я хочу {**функция программы**}, чтобы {**решаемая задача**}

История №1

Как разработчик, я хочу иметь ветвления и рекурсию, чтобы считать числа Фибоначчи

Критерии приёмки:

1. Есть if…then..else
2. Можно объявить и вызвать функцию
3. Есть поддержка рекурсии

История №1

Как разработчик, я хочу иметь ветвления и рекурсию, чтобы считать числа Фибоначчи

Критерии приёмки:

1. Ес… if…then..else
2. Можно объя… и вызвать функцию
3. Есть поддержка рекурс…

История №1

Как разработчик, я хочу иметь ветвления и рекурсию, чтобы считать числа Фибоначчи

Критерии приёмки:

1. Рекурсивное вычисление N-го числа Фибоначчи работает

Числа Фибоначчи — рекурсивная версия

```
def fib(n)
    if n < 1 then
        0
    else if n < 3 then
        1
    else
        fib(n-1) + fib(n-2);
```

Числа Фибоначчи — рекурсивная версия

```
def fib(n)
    if n < 1 then
        0
    else if n < 3 then
        1
    else
        fib(n-1) + fib(n-2);
```

Числа Фибоначчи — рекурсивная версия

```
def fib(n)
    if n < 1 then
        0
    else if n < 3 then
        1
    else
        fib(n-1) + fib(n-2);
```

Какова вычислительная сложность?

Числа Фибоначчи — итеративная версия

```
def fib(n)
    # ... if n < 3
    else var a = 1, b = 1, c in (
            for i = 2, i < n in
                c = a + b :
                a = b :
                b = c
        ) : b;
```

Числа Фибоначчи — итеративная версия

```
def fib(n)
    # ... if n < 3
    else var a = 1, b = 1, c in (
            for i = 2, i < n in
                c = a + b :
                a = b :
                b = c
        ) : b;
```

Какова вычислительная сложность?

Улучшаем историю

Как разработчик, я хочу иметь **ветвления, рекурсию и циклы**, чтобы считать числа Фибоначчи

Критерии приёмки:

1. **Рекурсивное** вычисление N-го числа Фибоначчи работает
2. **Итеративное** вычисление N-го числа Фибоначчи работает

Улучшаем историю

Как разработчик, я хочу иметь **ветвления, рекурсию и циклы**, чтобы считать числа Фибоначчи

Критерии приёмки:

1. **Рекурсивное** вычисление N-го числа Фибоначчи работает
2. **Итеративное** вычисление N-го числа Фибоначчи работает

Подход ATDD

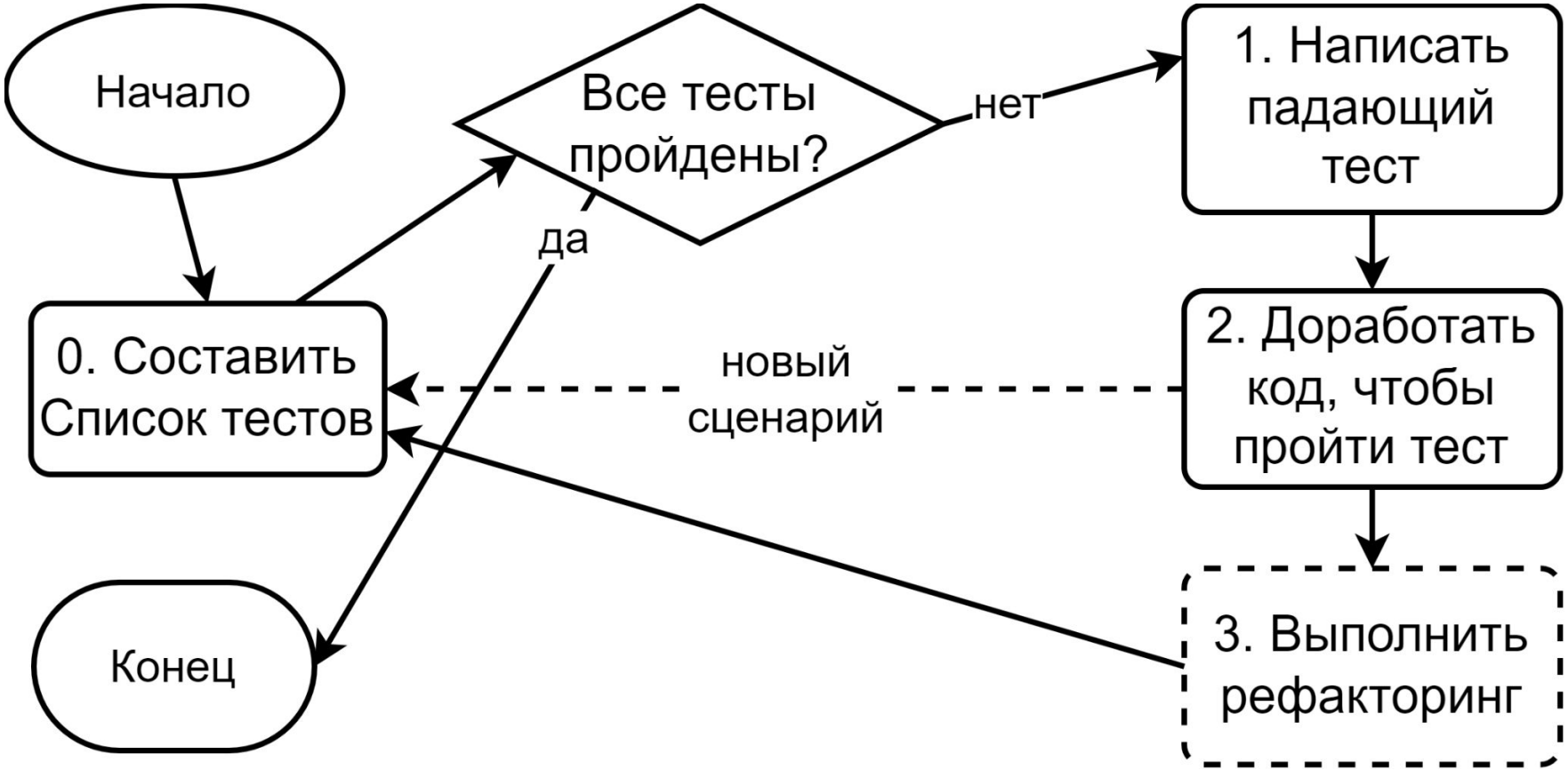
Проблема

Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	Написать 1. Пользовательскую историю 2. Критерии приёмки
Как понять, что критерии приёмки пройдены?	???

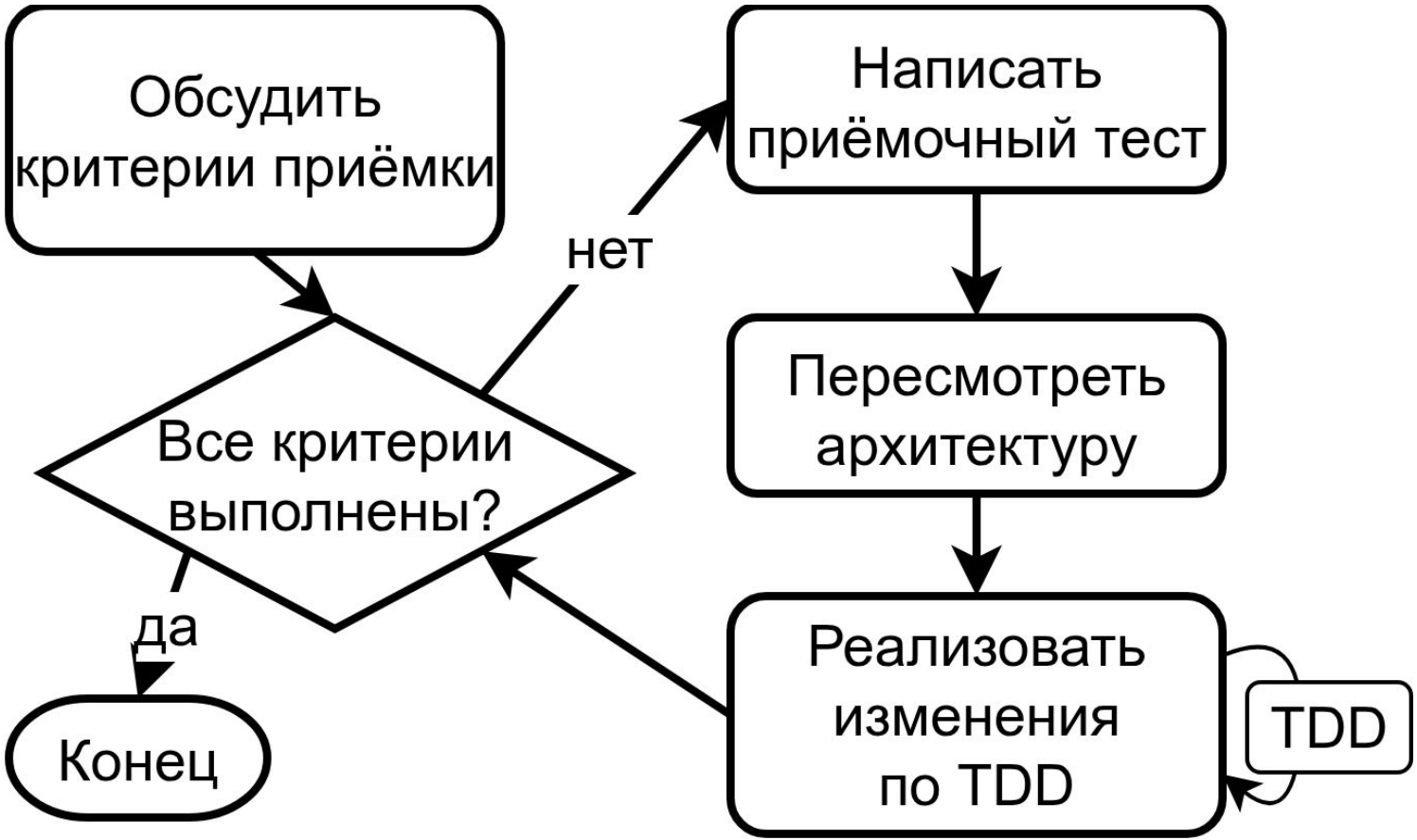
Проблема

Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	Написать 1. Пользовательскую историю 2. Критерии приёмки
Как понять, что критерии приёмки пройдены?	Написать приёмочный тест

TDD — Test Driven Development



ATDD — Acceptance Test Driven Development



ATDD — Acceptance Test Driven Development

Один день по ATDD:

9:30 — пишем приёмочный тест

10:00 — рефакторим архитектуру

14:00 — реализуем изменения

17:00 — тест пройден,

закрываем задачу

ATDD — Acceptance Test Driven Development

Один день по ATDD:

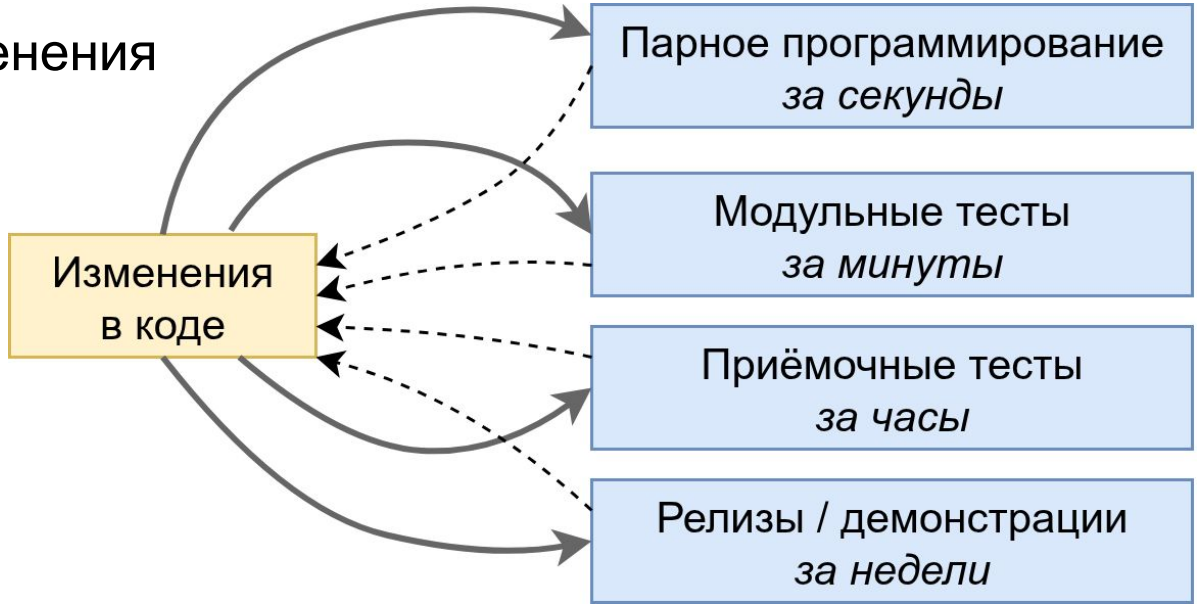
9:30 — пишем приёмочный тест

10:00 — рефакторим архитектуру

14:00 — реализуем изменения

17:00 — тест пройден,

закрываем задачу



Ветвления и циклы

Грамматика выражений

```
primary_expression = number
    | identifier
    | identifier, arguments_list,
    | if_else_expression
    | for_loop_expression
    | variable_definition_scope
    | "(", expression, ")" ;
```

Выражения
if … then … else
for … in

Выражения if…then…else

```
if_else_expression =
  "if", expression,
  "then", expression,
  "else", expression ;
```

Выражения if…then…else

```
if_else_expression =
  "if", expression,
  "then", expression,
  "else", expression ;
```

Проблема «висячего else»:

```
if ... then
    if then ...
else ...
```

- К какому if относится else?
- Возникает ли проблема в Kaleidoscope?

Разбор if…then…else

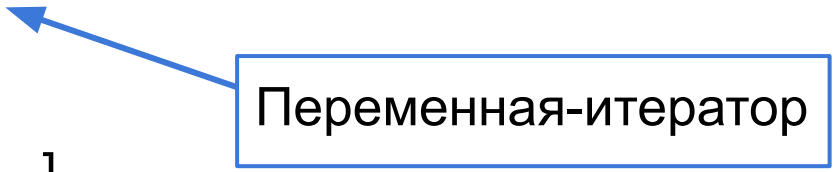
```
private Expression ParseIfElseExpression() {
    Match(TokenType.If);
    Expr c = ParseExpression();
    Match(TokenType.Then);
    Expr t = ParseExpression();
    Match(TokenType.Else);
    Expr e = ParseExpression();
    return new IfElseExpr(c, t, e);
}
```

Выражения for…in

```
for_loop_expression =
  "for", identifier, "=", expression,
  ",", expression,
  [ ",", expression ],
  "in", expression ;
```

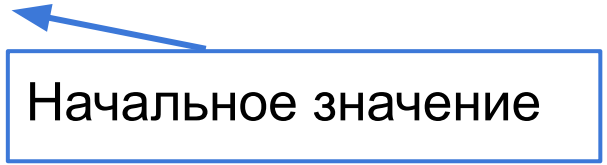
Выражения for…in

```
for_loop_expression =
  "for", identifier, "=", expression,
  ",", expression,
  [ ",", expression ],
  "in", expression ;
```



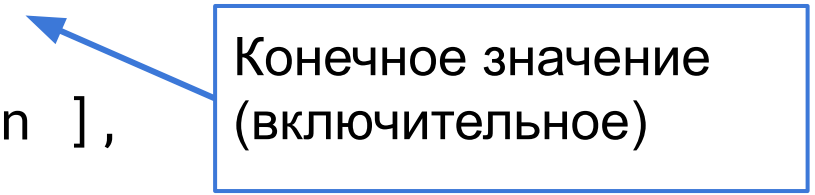
Выражения for…in

```
for_loop_expression =
  "for", identifier, "=", expression,
  ",", expression,
  [ ",", expression ],
  "in", expression ;
```



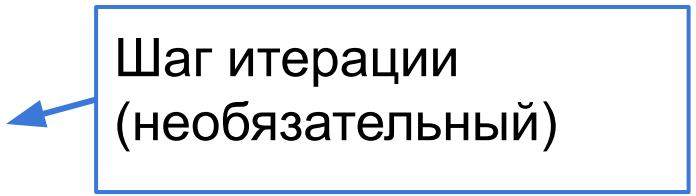
Выражения for…in

```
for_loop_expression =
  "for", identifier, "=", expression,
  ",", expression,
  [ ",", expression ],
  "in", expression ;
```



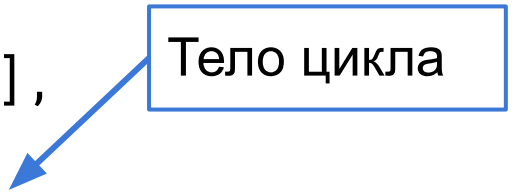
Выражения for…in

```
for_loop_expression =
  "for", identifier, "=", expression,
  ",", expression,
  [ ",", expression ],
  "in", expression ;
```

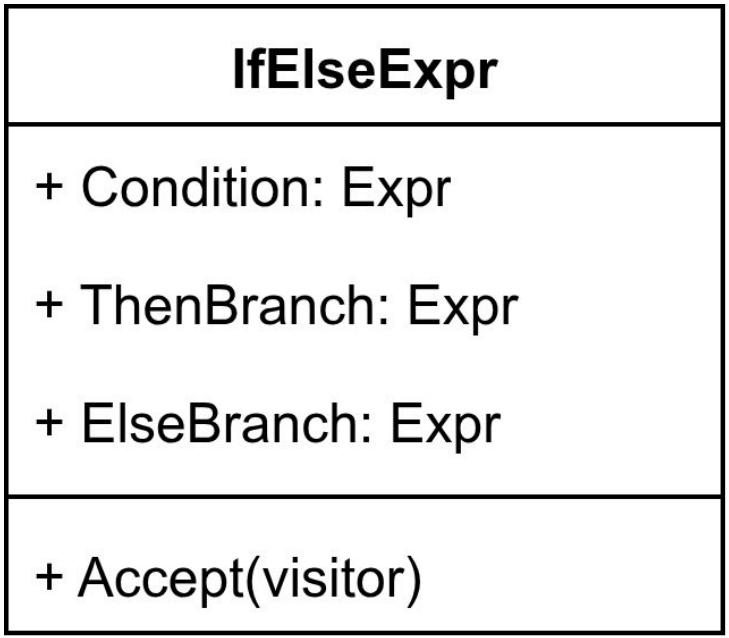


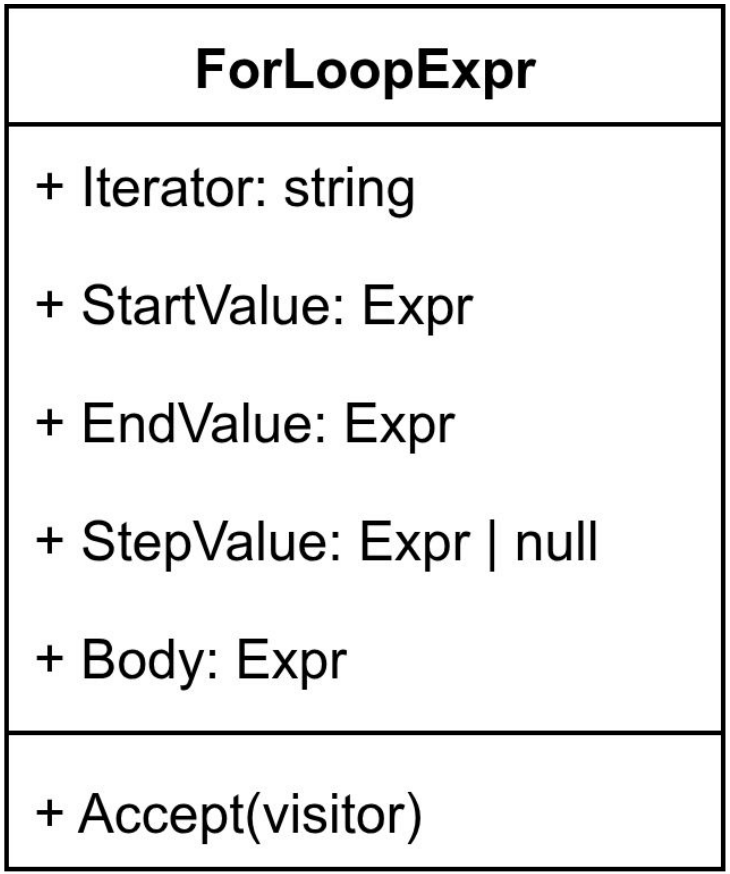
Выражения for…in

```
for_loop_expression =
  "for", identifier, "=", expression,
  ",", expression,
  [ ",", expression ],
  "in", expression ;
```



Изменения в AST





Условное выполнение

```
public void Visit(IfElseExpression e) {
    e.Condition.Accept(this);
    double value = _values.Pop();
    if (!Numbers.AreEqual(0.0, value)) {
        e.ThenBranch.Accept(this);
    } else  {
        e.ElseBranch.Accept(this);
    }
}
```

Аналитикам на подумать

1. if-then-else — выражение или инструкция?
    - проблема висячего else
2. switch — есть или нет?
    - проблема пропущенного break — fallthrough
3. циклы — выражение или инструкция?
    - формы циклов — while, do…while, for, foreach
    - инструкции break / continue

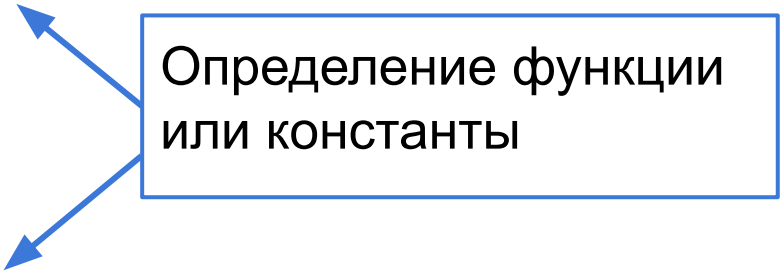
Пользовательские функции

Грамматика выражений

```
primary_expression = number
    | identifier
    | identifier, arguments_list,
    | if_else_expression
    | for_loop_expression
    | variable_definition_scope
    | "(", expression, ")" ;
```

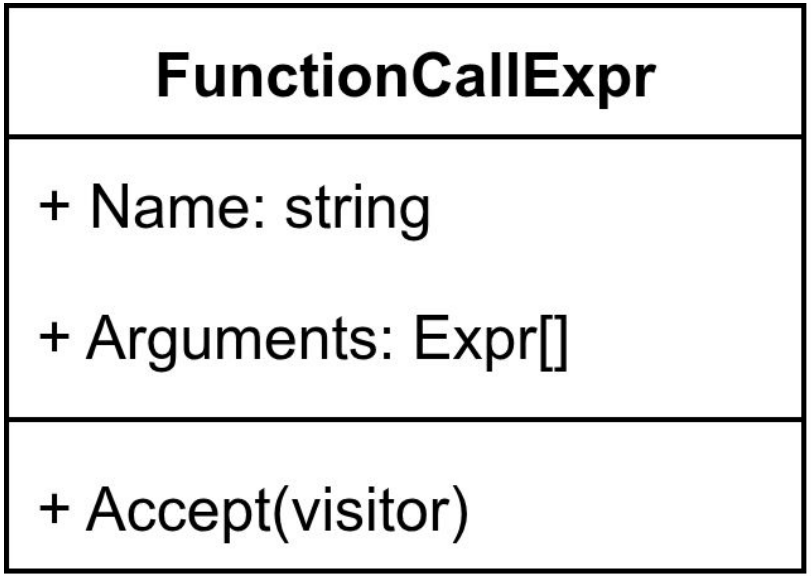
Грамматика программы

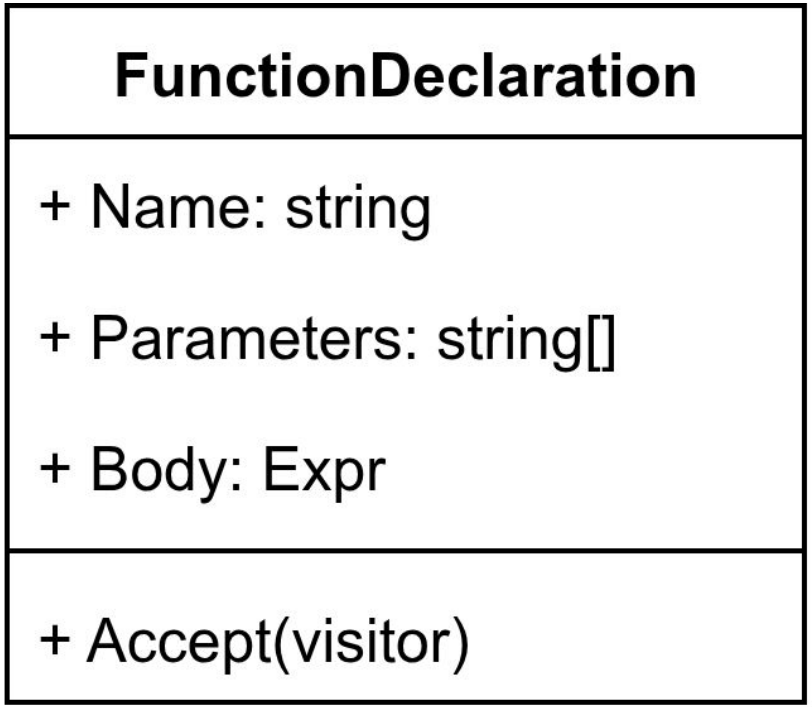
```
top_level_statement = (
    function_or_constant_definition
    | expression
), [ ";" ] ;
```




```
function_or_constant_definition = "def", identifier
    [ , "(", parameter_list, ")" ], expression ;
```

Изменения в AST





Разбор определения

```
private AstNode ParseFunctionConstantDefinition() {
    Match(TokenType.Def);
    string name = Match(TokenType.Ident).Value!.ToString();
    if (_tokens.Peek().Type == TokenType.OpenParenthesis) {
        return new FunctionDecl(
          name, ParseParameterList(), ParseExpr()
        );
    } else {
        return new ConstantDecl(name, ParseExpr());
    }
```

Вычисление определения функции

```
public void Visit(FunctionDeclaration d) {
    _context.DefineFunction(d);
    // NOTE: Результат «вычисления» — число 0.0.
    _values.Push(0.0);
}
```

Аналитикам на подумать

1. Есть ли функции без параметров?
2. Пересекаются ли имена функций и переменных?
3. Инструкция return
4. Рекурсия
5. Взаимная рекурсия
6. Порядок вычислений аргуменов

Подытожим

Проблемы / решения из прошлой лекции

Проблема	Решение
Циклы и функции требуют повторного выполнения	Внедряем AST (Abstract Syntax Tree)
Рефакторинг может сломать программу	Внедряем приёмочные тесты (функциональные тесты)
Как отделить тесты от реализации?	Используем Gherkin и Cucumber / Reqnroll

Проблемы / решения из этой лекции

Проблема	Решение
Как понять, что функции, циклы и ветвления реализованы верно?	Написать 1. Пользовательскую историю 2. Критерии приёмки
Как понять, что критерии приёмки пройдены?	Написать приёмочный тест

Спасибо за внимание!
Вопросы?