Содержание

1 Описание синтаксиса и семантики компилируемого языка

Так как Ruby – язык с динамической типизацией, то определенных типов переменных нет. Ограничение на использование переменных – переменная объявляется только при присвоении ей какого-либо значения. Если до этого переменной не было присвоено ни одного значения, то оно равно nil. Пример присвоения: a=5.

Запись логических операторов указана в таблице 5.

Командами ввода/вывода являются функции языка – gets()/puts().

Ветвления в языке представлены следующим синтаксисом:

```
 if (a < 5)
   puts(“a<5”)
elsif (a>5)
   puts("a>5")
else
   puts("a=5")
end
```

Циклы в языке представлены следующим синтаксисом:

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
for i in a
   puts(i)
end

while (i < 15)
   i=i+1
end
```

Числовые константы в языке представлены с помощью целых чисел и чисел с плавающей точкой. У последних возможна запись, как в экспоненциальной форме – разделителем тогда служит символ e латиницей, так и через точку, отделяя целую часть от дробной. Целые числа можно записывать в двоичной, восьмеричной, десятичной, шестнадцатеричной системах.

Примеры: 4; 5.5; 4e8; 0b1010; 0xF1B; 0285.

Логические константы представлены в языке значениями true (истина) и false (ложь).

Символьные и литеральные константы могут быть записаны четырьмя методами, указанными в Таблице 5. При этом в случае двойных кавычек внутри без экранирования могут быть использованы одинарные, двойные – с экранированием. В случае одинарных – двойные без экранирования, одинарные – с экранированием. %q действует как одинарные кавычки, %Q – как двойные.

Таблица 1. Арифметические операции языка

сложение	x + y
вычитание	x – y
умножение	x * y
деление	x / y

Таблица 2. Операции сравнения языка

равно	x == y
не равно	x != y
меньше	x<y
больше	x>y

Таблица 3. Операции для работы со строками

сложение строк	str+str1
умножение строк на целое число	str1*3

Комментарии в языке могут быть однострочными и многострочными. Однострочные располагаются в любом месте строки и начинаются с символа #. Начало и конец многострочного комментария должны быть расположены в начале строки.

Методы разграничиваются ключевыми словами def и end. После def следует имя метода и аргументы в круглых скобках. По умолчанию в методах возвращается результат последнего вычисленного выражения.

Пример метода:

```
def t(z,c)
        puts(z)
        puts(c)
end
```

Классы в языке создаются с использованием ключевого слова class, за которым следует имя класса с большой буквы. Методы класса объявляются внутри тела класса, как обычные функции. Поля класса обозначаются как @a, а статические переменные как @@ b, и могут использоваться или внутри класса по имени, или через геттеры/сеттеры.

Пример класса:

```
class D
        def  getA
                puts(@a)
        end
end
```

Создание объекта происходит через метод new, который вызывает конструктор класса с соответствующим числом аргументов.

2 Пример программы с использованием всех реализованных возможностей

```
#Создаем массив
list=[1,2,3,4,5,6,7,8,8,0]
#Изменяем предпоследний элемент
list[-2]=9
#Печатаем диапазоны
puts(list[0...5])
puts(list[5..9])

#Создание строк
str1="\"'String'\""
str2="'\'String\'"
str3=%q!"'String"'!
str4=%Q{"'String"'}

#Создание целых значений
int1=1_000_000
int2=0x1E
int3=0b100
int4=-0453

#Ветвления
if int1<int2
   puts("in if")
elsif(int2>int3)
   puts("in elsif")
else
   puts("in else")
end

#Циклы
while(int2>int3)
   int2=int2-3
end

for i in list[1..4]
   puts(i)
end

#Процедуры
```

```
def g
   x=[1,2,3]
   for i in x
      puts(i)
   end
   return "Ok"
end

def f(a,b)
   c=a+b
end

#Ввод и вывод
puts(gets())

#Дробные числа
float1=2.5
float2=2e5

#Операции
puts(list[1]+list[4])
puts(1+2.5)
puts(5/2.5)
puts(16.4-2.9)
puts([1]+[3,4,5])

#Классы

class A
   def initialize(a,b)
      @@a=a
      @b=b
   end
   def a(a,b)
      return a+b
   end
end
b=A.new(4,9)
puts(b.a(2,3))
```

3 Таблица разделения обязанностей в команде

Таблица 4. Разделение обязанностей в команде

Возможность	Вес	Особенности в языке	Кто реализует
Локальные переменные встроенных типов данных : • целые числа (один тип), • символы • строки Выражения с использованием локальных переменных, арифметических операций (4 вида), операций сравнения и присваивания	4 min	Динамическая типизация	Островский
Одномерные массивы (или контейнеры), операция доступа к элементу массива	3 min	• Массивы можно создать с помощью [ ] • Можно получить элемент по отрицательному индексу (отсчет с конца массива). • Можно получить элементы по диапазону.	Островский

Продолжение Таблицы 4

Возможность	Вес	Особенности в языке	Кто реализует
Числовые (целые числа), символьные и строковые константы (литералы) с поддержкой всех видов констант и служебных последовательностей символов	2 min	• Строки записываются в двойных или одинарных кавычках. • В одинарных кавычках экранируются только одинарная кавычка и символ обратной косой черты. Если в строке присутствуют мета-символы, то они автоматически экранируются. • В двойных кавычках необходимо экранирование всех мета-символов. • В строках, содержащих большое количество метасимволов можно воспользоваться конструкциями %q и %Q, чтобы не экранировать встречающиеся в строке кавычки. Вслед за ними должна идти строка, обрамленная с обеих сторон символами-ограничителями (любой символ, не являющийся буквенно-цифровым). При этом %q ведет себя как одиночные кавычки, а %Q как двойные. • В середине чисел могут находиться единичные символы нижнего подчеркивания, которые игнорируются • Если перед числом пишется 0x – признак того, что число записано в шестнадцатеричной	Шалиевская

Продолжение Таблицы 4

Возможность	Вес	Особенности в языке	Кто реализует
		системе. • Если перед числом пишется 0b – признак того, что число записано в двоичной системе • Если перед числом пишется 0 – признак того, что число записано в восьмеричной системе. • Допустимые метасимволы: \t, \r, \a, \b, \s, \v, \f	
Управляющие структуры: развилки	3 min	• Условие выполнения не обязательно ставить в скобки.	Островский
Управляющие структуры: циклы	3 min	• Цикл for может проходить по заданному диапазону или по элементам массива • Условие в цикле while можно писать со скобками или без	Шалиевская
Функции (процедуры)	3 min	• Начинаются ключевым словом def, заканчиваются ключевым словом end • В вызове и объявлении метода без аргументов можно опускать круглые скобки • Если return не прописан пользователем, из метода возвращается результат последнего вычисленного выражения или nil	Шалиевская

Продолжение Таблицы 4

Возможность	Вес	Особенности в языке	Кто реализует
Классы/функции для работы с консолью (ввод/вывод базовых типов данных)	2 min	• Для вывода на экран используется puts Для ввода данных используется gets(можно не писать скобки)	Шалиевская
Переменные, константы и операции (арифметические, сравнения, присваивания) для чисел с плавающей точкой	3	• Могут быть записаны в экспоненциальной форме. Если в операции присутствует хотя бы одно вещественное число, то и результат будет вещественным числом..	Островский
Классы, свойства и методы, одиночное наследование с динамическим связыванием	4	• Идентификатор класса должен начинаться с заглавной буквы • Класс определяется при помощи ключевого слова class, за которым следует end • В классе можно создавать конструкторы с именем initialize • В любом методе можно вызывать метод родителя с таким же именем с помощью super(можно вызывать метод, где другое количество аргументов) • Текущий объект обозначается словом self • Поля класса обозначаются как @id • Внутри класса к полям можно обращаться по имени или через сеттер	Шалиевская

Продолжение Таблицы 4

Возможность	Вес	Особенности в языке	Кто реализует
		• Вне класса к полям можно обращаться только через специальные методы • Объект класса создается с помощью метода new, который может не принимать аргументов(быть по умолчанию) или принимать аргументы конструктора, который создал пользователь	
Вложенные классы		• Поддерживаются вложенные классы • Если класс вложен в другой, и при этом наследуется от него, то дочерний видит все методы родителя, даже если они объявлены после самого класса	Шалиевская
Статические поля	1	• Статические поля класса обозначаются как @@id • К статическим полям класса можно обращаться только внутри методов по имени • За пределами класса обращаться с помощью специальных методов	Шалиевская

4 Таблица лексем

Таблица 5. Лексемы языка

Вид лексемы	**Лексема**	**Семантическое значение**	**Описание**
key word	While		Ключевое слово цикла while
key word	For		Ключевое слово цикла for
key word	In		Ключевое слово цикла for
key word	If		Ключевое слово условного выражения
key word	Elsif		Ключевое слово условного выражения
key word	Else		Ключевое слово условного выражения
key word	True		Ключевое слово (логическая константа)
key word	False		Ключевое слово (логическая константа)
key word	Nil		Ключевое слово (означает «отсутствие значения»)
key word	Self		Ключевое слово (ссылка на текущий объект)
key word	Return		Ключевое слово возврата результата
key word	Class		Ключевое слово объявления класса
key word	Def		Ключевое слово объявления функции
key word	End		Ключевое слово завершения тела циклов, условий, объявления функций и классов

Продолжение Таблицы 5

Вид лексемы	**Лексема**	**Семантическое значение**	**Описание**
compare operation	<, >, ==, !=		Операторы сравнения
syntax	[		Открывающаяся квадратная скобка
syntax	]		Закрывающаяся квадратная скобка
syntax	(		Открывающаяся круглая скобка
syntax	)		Закрывающаяся круглая скобка
syntax	,		Запятая
syntax	.. …		Диапазоны массивов
arithmetic operation	=, *, /, +, -		Арифметические операции
identifier	[A-Za-z_][A-Za-z_0-9]*	Имена переменных и функций состоят только из букв, цифр и ‘_’, начинаются НЕ с цифры	Название переменных и функций
identifier	@[A-Za-z_][A-Za-z_0-9]*	Имена полей класса состоят из ‘@’ и имени переменной	Название полей класса
identifier	@@[A-Za-z_][A-Za-z_0-9]*	Имена статических переменных класса состоят из “@@” и имени переменной	Название статических переменных класса
constant	«строка», ‘строка’, %q{строка}, %Q{строка}	Строки 4х видов.	Константы

Продолжение таблицы 5

Вид лексемы	**Лексема**	**Семантическое значение**	**Описание**
Constant	Целое число	Целые числа состоят из цифр и ‘_’. Могут быть записаны в двоичной, восьмеричной, десятичной, шестнадцатеричной формах	Константы
Constant	Числа с плавающей точкой	Дробные числа состоят из цифр и ‘.’, которая отделяет целую часть от дробной. Может записываться в экспоненциальной форме	Константы
Comment	#.....		Однострочный комментарий
Comment	=begin… … =end		Многострочный комментарий

5 Грамматика языка

```
program: stmt_list
;
stmt_list: stmt
| stmt_list stmt
;
stmt: ENDL
| expr ENDL
| ifStmt ENDL
| whileStmt ENDL
| forStmt ENDL
| funcDefStmt ENDL
| classStmt ENDL
;
stmt_list_forClass: stmt_forClass
| stmt_list_forClass stmt_forClass
;
stmt_forClass: ENDL
| funcDefStmt ENDL
| classStmt ENDL
;
lowerStmt_list: lowerStmt
| lowerStmt_list lowerStmt
;
lowerStmt: ENDL
| expr ENDL
| ifStmt ENDL
| whileStmt ENDL
| forStmt ENDL
| RETURN expr ENDL
;
expr: IINT
| FFLOAT
| STR
| FFALSE
| TTRUE
| NIL
| ID
| INST_ID
| ID '(' factArgListEmpty ')'
| SELF
| '(' expr ')'
| expr '+' expr
| expr '-' expr
| expr '*' expr
| expr '/' expr
| expr '<' expr
| expr '>' expr
| expr '=' expr
| expr EQSPACES expr
```

```
| expr EQ expr
| expr NE expr
| '-' expr %prec UMINUS
| expr '[' expr DIAPAZONIN expr ']'
| expr '[' expr DIAPAZONOUT expr ']'
| expr '[' expr ']'
| expr '.' ID
| expr '.' INST_ID
| expr '.' ID '(' factArgListEmpty ')'
|'[' factArgList ']'
;
factArgListEmpty: /*empty*/
| factArgList
;
factArgList: expr
| factArgList ',' expr
;
formalArgListEmpty: /*empty*/
| formalArgList
;
formalArgList: ID
| formalArgList ',' ID
;
ifStmt: IF expr ENDL lowerStmt_list END
| IF expr ENDL lowerStmt_list ELSE lowerStmt_list END
| IF expr ENDL lowerStmt_list elsif_list END
| IF expr ENDL lowerStmt_list elsif_list ELSE lowerStmt_list END
;
elsif_list: ELSIF expr ENDL lowerStmt_list
| elsif_list ELSIF expr ENDL lowerStmt_list
;
forStmt: FOR ID IIN expr ENDL lowerStmt_list END
| FOR INST_ID IIN expr ENDL lowerStmt_list END
| FOR INST_ID IIN expr ENDL END
| FOR ID IIN expr ENDL END
;
whileStmt: WHILE expr ENDL lowerStmt_list END
| WHILE expr ENDL END
;
funcDefStmt: DEF ID '(' formalArgListEmpty ')' lowerStmt_list END
| DEF ID '(' formalArgListEmpty ')' END
| DEF ID ENDL lowerStmt_list END
| DEF ID ENDL END
| DEF ID '=' '(' formalArgListEmpty ')' lowerStmt_list END
| DEF ID '=' '(' formalArgListEmpty ')' END
;
classStmt: CLASS ID stmt_list_forClass END
| CLASS ID END
| CLASS ID '<' ID ENDL stmt_list_forClass END | CLASS ID '<' ID ENDL
END
;
```

6 Перечень таблиц, генерируемых на этапе семантического анализа с описаниями структур, описывающих строчки таблицы

а) Структура, хранящая информацию о классах

```
struct tables
{
	int papa; // ссылка на константу Class родителя
	int constr; // номер константы methodRef родительского конструктора по умолчанию
	int done; // флаг, закончен ли просмотр кода класса
	std::vector<function> functions; // список всех функций в классе
	std::vector<std::string> classField; // список всех полей класса
	std::vector<int> numberField; // номер констант fieldRef полей класса
	std::vector<std::string> classVar; // список всех статических переменных
	struct ClassStruct* ClassSt; // структура этого класса
	std::vector<std::string> classesIsee; //классы о которых знает текущий класс
	std::vector<std::string> classesExternals;// внешние классы для текущего класса
	std::vector<std::string> classesIn; // вложенные в текущий классы
	std::vector<std::string> flyLocal; // переменные, лежащие в классе, но не в методе (только для main класса def)
	std::vector<std::pair<function,std::string>> functionsIsee;// функции, которые видно в текущем классе(например функции родителя)
	std::vector<constTable> tableConstant; // таблица констант класса
	int strNumMethod;// сколько методов
	int strNumLocal;//сколько «flyLocal»
	int strNumConst;//сколько записей в таблице констант
};
```

б) Структура, хранящая информацию о функциях

```
struct function
{
	std::string functionName; // имя функции
	int argCount; // число аргументов в функции
	std::vector<std::string> localVarList; // список всех переменных
	int str; //номер последней строки в таблице локальных переменных
	FuncDefStruct* funcDef; //структура функции
	int NameNumber; // ссылка на константу с Utf-8 с именем класса
	int deskNumber; // ссылка на константу с Utf-8 с дескриптором класса
};
```

в) Структура записи таблицы констант

```
struct constTable
{
	int num; // номер строки
	std::string type; // тип константы
	std::string value; // значение константы
};
```

7 Перечень ошибок, определяемых на этапе семантического анализа

Таблица 6. Описание ошибок и сообщения об их обнаружении

Описание	Сообщение об ошибке
Return вызывается вне функции	`Return is not in function`
Левой части операции присвоения невозможно присвоить значение	`Assign without operand`
Присвоение совершается методу	`Can not assign a value to a method`
Использование статической переменной вне класса	`Field without class`
Использование непроинициализованной переменной	`Can't use undefined operand`
Нельзя использовать присвоение внутри выражения индекса при взятии элемента списка(массива) по индексу	`Can't use assignment operation in expression for array index.`
Использование поля класса вне класса	`Field without class`
Использование необъявленной функции	`Function not assign`
Использование метода super внутри класса без родителя	`No parent for super`
Использование метода super внутри класса, где у родителя этого класса данного метода нет	`No parent method`
Вызов метода super вне метода	`Super must be in method`
Вызов метода super вне класса	`Super called outside class`
Вызов конструктора вне метода класса	`Initialize can be only in method def`
Создание объекта с использованием несуществующего конструктора	`No initialize method`
Создание объекта несуществующего класса	`No class`
Обращение к полю класса через self или какой-то объект	`You can not get a field from this object`

Продолжение таблицы 7

Описание	Сообщение об ошибке
Вызов метода для несуществующего объекта	`Object not assign`
Использование непроинициализованного массива	`Arr not assign`
В объявление функции одинаковые имена аргументов	`Arg name use already`
Объявление уже существующей функции	`Function name use already`
Объявление уже существующего класса	`Class name use already`
Класс, указанный как родитель, не существует	`Parent class not exist`

8 Перечень преобразований дерева и дополнительной информации для узлов дерева на этапе семантического анализа

В ходе этапа семантического анализа была преобразована операция присвоения элементу массива к тернарному виду. В функции, где пользователь не указал возврат значения, было добавлено возвращение результата последнего вычисленного выражения или nil. Условия типа elif (else if) были преобразованы во вложенные. Обращение к полям класса было заменено на обращение к полю объекта self. Если пользователь в своих классах не создавал конструкторы – добавлялись конструкторы по умолчанию. Также было проведено атрибутирование дерева – на узлы операндов были вынесены номера в таблицах констант и локальных переменных. Также на узлы объявления и вызова функции были добавлены номера ссылок на соответствующие методы класса.

Построим дерево по следующему коду:

```
class A
     def setA(value)
          @a=value
     end
end
c=A.new
z=[c]
z[0].setA(2)
```

Пример атрибутированного дерева представлен на рис. 1.

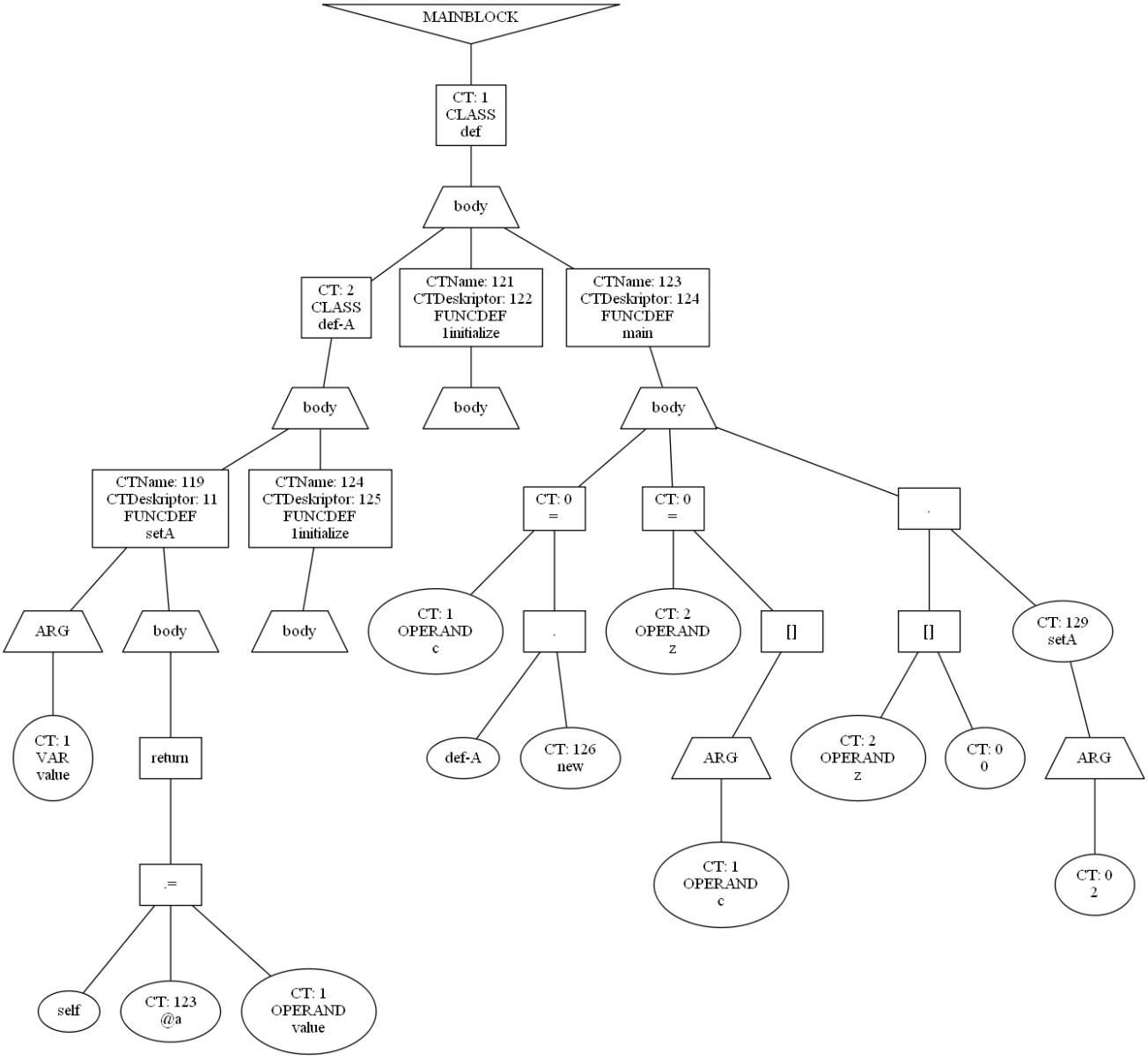

Рисунок 1. Пример атрибутированного дерева

Выводы

В ходе данной работы был создан компилятор для языка Ruby (v 2.4.1), генерирующий код языка верхнего уровня в Java Byte Code. Поскольку компилятор создавался в несколько этапов, нацеленных на каждый уровень анализа программы, были освоены методики лексического, синтаксического и семантического анализа. Кроме того была изучена работа с виртуальной машиной Java и управление ею с помощью bytecode.