**Язык для помощи в работе со строками (Straid)**

**Спецификация Straid: синтаксис объявления переменных и подпрограмм**

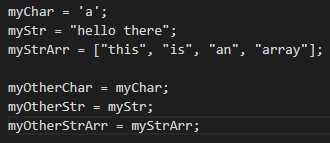
В языке Straid четыре встроенных типа данных: **int** (неотрицательное целое число, либо число -1) **char** (символ), **string** (строка) и **strarray** (массив строк). Переменные и константы типа **char** используются для представления единственного символа. Поддерживаются только символы таблицы ASCII; далее по документу под словом “символ” будут подразумеваться именно такие символы. Переменные и константы типа **string** представляют собой упорядоченные пронумерованные последовательности констант типа **char**, нумерация элементов строк начинается с 0. Переменные и константы типа **strarray** представляют собой упорядоченные пронумерованные последовательности констант типа **string**, нумерация элементов начинается с 0.

Константы типа **char** имеют вид символа, заключённого в одиночные кавычки (‘a’, ‘b’, ‘c’, ‘0’, ‘ ’, …).

Константы типа **string** имеют вид последовательности символов, заключённой в двойные кавычки (“string”, “hello”, “this is a string with spaces”, …). Максимальная возможная длина строки составляет 65535 символов. Попытка получения строки большей длины любыми способами породит ошибку.

Константы типа **strarray** имеют вид последовательности констант типа **string**, разделённых запятыми. Сама последовательность при этом заключена в квадратные скобки ([“this”, “is”, “an array”, “of strings”]). При желании пробельные символы между символом запятой и очередной строкой массива строк могут быть опущены.

Straid использует **неявное объявление переменных**. Объявление переменных производится путём сопоставления идентификатора переменной с её значением с помощью оператора присваивания (=). Значением переменной при её объявлении может выступать как константа, так и другая переменная. Во втором случае значение другой переменной будет скопировано в объявляемую переменную. Ниже представлены примеры корректного объявления переменных



Оператор присваивания в Straid **многоцелевой** и может использоваться для объявления нескольких переменных с использованием одной строки кода:

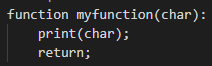


Переменные можно объявлять и без начального значения. Для этого в строке пишется лишь название объявляемой переменной:

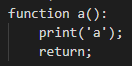


Переменная, заданная таким образом, не содержит в себе никакого значения. Попытка использовать такую переменную в какой-либо операции, а также попытка передать такую переменную в подпрограмму в качестве параметра, породит ошибку. В случае, если переменная с таким именем уже объявлена в текущей области видимости (об областях видимости позже), её значение будет утеряно.

Straid поддерживает работу пользовательских **подпрограмм**. Пользовательские подпрограммы объявляются ключевым словом **function**. Ниже представлен пример корректного объявления пользовательской подпрограммы.



При объявлении подпрограммы в скобках после названия подпрограммы через запятую указываются **аргументы**, которые могут передаваться в эту подпрограмму при её вызове основной программой или другими подпрограммами. В области видимости функции неявно создаются переменные с названиями, идентичными указанным в скобках при объявлении подпрограммы. В эти переменные копируются значения переданных функции аргументов в соответствующем порядке. Если объявляемая подпрограмма не должна принимать аргументов извне, следует оставить скобки пустыми:



Все пользовательские подпрограммы должны быть объявлены перед началом программы. Началом программы считается первая строка файла, не начинающаяся с табуляции и не содержащая в себе ключевого слова function. **Объявление подпрограммы внутри другой подпрограммы не допускается**.

Каждая объявляемая подпрограмма должна заканчиваться ключевым словом **return**. Если значение, полученное внутри подпрограммы, требуется для дальнейшего выполнения основной программы или другой подпрограммы, то после ключевого слова return пишется идентификатор переменной, содержащей это значение:



Пользовательские подпрограммы имеют свою собственную **область видимости**. Переменные, объявленные в пределах некоторой подпрограммы, могут быть использованы только в пределах блока кода, заключаемого этой подпрограммой. К переменным, “заключённым” в области видимости подпрограммы, относятся переменные, объявленные внутри этой подпрограммы, а также переменные, переданные подпрограмме в качестве аргументов, если таковые имеются.

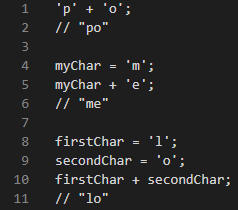
Язык Straid поддерживает комментарии. Строки-комментарии должны начинаться с двойного прямого слэша. “Закомментированные” строки будут игнорироваться компилятором.



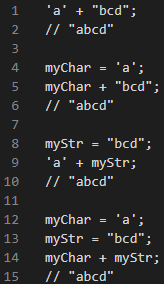
**Спецификация Straid: синтаксис операций над данными**

Синтаксис встроенных операций над данными:

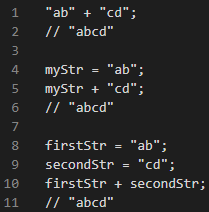
1. Бинарное сложение, +
   1. Сложение символов
      1. Синтаксис: **char1 + char2**, где char1 и char2 – переменные или константы типа **char**
      2. Возвращаемое значение: константа типа **string**, состоящая из символов char1 и char2 с сохранением порядка
      3. Примеры:



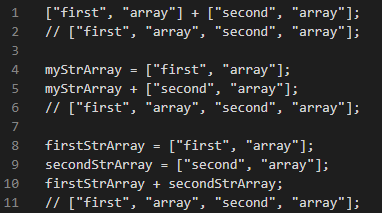
* 1. Сложение символа и строки
     1. Синтаксис: **char + str** или **str + char**, где char – переменная или константа типа **char**, str – переменная или константа типа **string**
     2. Возвращаемое значение: константа типа **string**, состоящая из символа char и символов строки str с сохранением порядка
     3. Примеры:



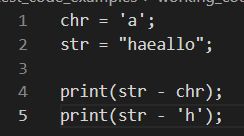
* 1. Сложение строк
     1. Синтаксис: **str1 + str2**, где str1 и str2 – переменные или константы типа **string**
     2. Возвращаемое значение: константа типа **string**, состоящая из символов строк str1 и str2 с сохранением порядка
     3. Примеры:



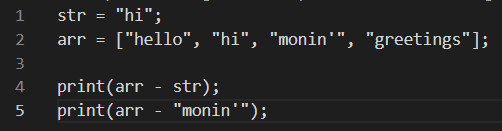
* 1. Сложение массивов строк
     1. Синтаксис: **strarray1 + strarray2**, где strarray1 и strarray2 – переменные или константы типа **strarray**
     2. Возвращаемое значение: константа типа **strarray**
     3. Примеры:



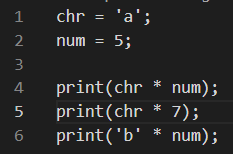
1. Бинарное вычитание, -
   1. Вычитание символа из строки
      1. Синтаксис: **str – char**, где char – переменная или константа типа **char**, str – переменная или константа типа **string**
      2. Возвращаемое значение: константа типа **string**, полученная путём удаления из str первого с начала строки символа, равного char; если в исходной строке такого символа не было, возвращает исходную строку
      3. Примеры:



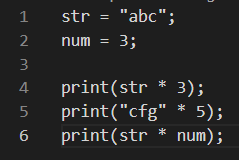
* 1. Вычитание строки из массива строк
     1. Синтаксис: **strarray – str**, где strarray – переменная или константа типа **strarray**, str – переменная или константа типа **string**
     2. Возвращаемое значение: константа типа **strarray**, полученная путём удаления из strarray первой с начала массива строки, равной str; если в исходном массиве такой строки не было, возвращает исходный массив
     3. Примеры:



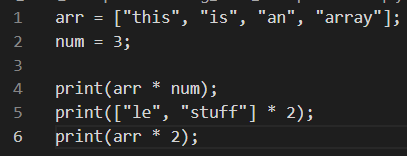
1. Бинарное умножение, \*
   1. Умножение символа на число
      1. Синтаксис: **char \* int**, где char – переменная или константа типа **char**, int – константа или переменная типа **int**
      2. Возвращаемое значение: константа типа **string**, состоящая из int символов char
      3. Примеры:



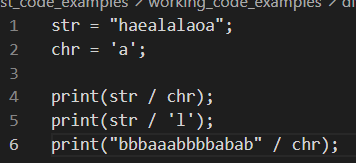
* 1. Умножение строки на число
     1. Синтаксис: **str \* int**, где str – переменная или константа типа **string**, int – константа или переменная типа **int**
     2. Возвращаемое значение: константа типа **string**, состоящая из int конкатенированных строк str
     3. Примеры:



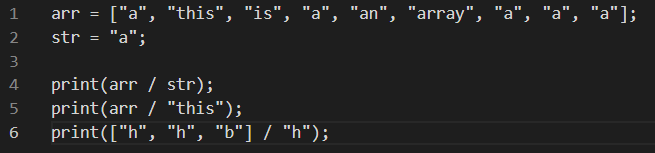
* 1. Умножение массива строк на число
     1. Синтаксис: **strarray \* int**, где strarray – переменная или константа типа **strarray**, int – константа типа **int**
     2. Возвращаемое значение: константа типа **strarray**, состоящая из int конкатенированных массивов strarray
     3. Примеры:



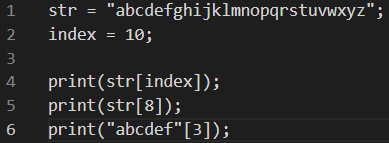
1. Бинарное деление, /
   1. Деление строки на символ
      1. Синтаксис: **str / char**, где char – переменная или константа типа **char**, str – переменная или константа типа **string**
      2. Возвращаемое значение: константа типа **string**, полученная путём удаления из str всех символов равных char; если в исходной строке таких символов не было, возвращает исходную строку
      3. Примеры:



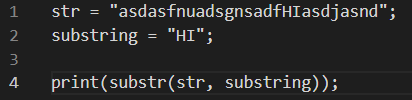
* 1. Деление массива строк на строку
     1. Синтаксис: **strarray / str**, где strarray – переменная или константа типа **strarray**, str – переменная или константа типа **string**
     2. Возвращаемое значение: константа типа **strarray**, полученная путём удаления из strarray всех строк, равных str; если в исходном массиве таких строк не было, возвращает исходный массив
     3. Примеры:



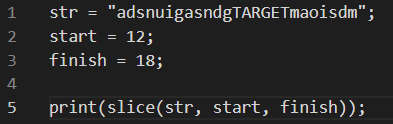
1. Оператор доступа по индексу, []
   1. Синтаксис: **str[int]**, где str – переменная или константа типа **string**, int – константа или переменная типа **int**
   2. Возвращаемое значение: константа типа **char**, равная символу, расположенному в строке str по индексу int
   3. Примеры:



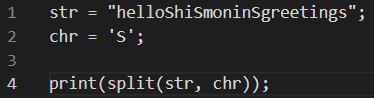
1. Функция поиска подстроки, substr()
   1. Синтаксис: **substr(str, substring)**, где str, substring – переменные или константы типа **string**
   2. Возвращаемое значение: константа типа **int**, равная индексу начала подстроки substring в строке str; если подстроки substring нет в строке str, возвращает -1
   3. Примеры:



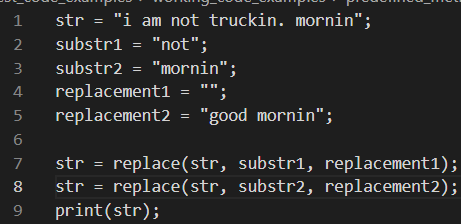
1. Функция извлечения среза, slice()
   1. Синтаксис: **slice(str, begin, end)**, где str – переменная или константа типа **string**, begin, end – переменные или константы типа **int**
   2. Возвращаемое значение: константа типа **string**, полученная путём извлечения из str подстроки, начинающейся с индекса begin и заканчивающаяся индексом end не включительно; исключительные ситуации (begin < 0, end больший либо равный длине строки, begin >= end) порождают ошибку
   3. Примеры:



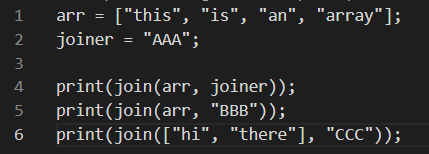
1. Функция разбиения, split()
   1. Синтаксис: **split(str, char)**, где char – переменная или константа типа **char**, str – переменная или константа типа **string**
   2. Возвращаемое значение: константа типа **strarray**, содержащая строки, полученные путём разделения строки str по символу char; в случае, если в строке str нет символа char, возвращает константу типа **strarray** с единственным элементом – строкой str
   3. Примеры:



1. Функция замены по шаблону, replace()
   1. Синтаксис: **replace(str, replacing, replacement)**, где str, replacing, replacement – переменные или константы **string**
   2. Возвращаемое значение: константа типа **string**, со всеми подстроками, совпадающими с replacing, заменёнными на replacement; если подстрок replacing в исходной строке не было, возвращает исходную строку
   3. Примеры:



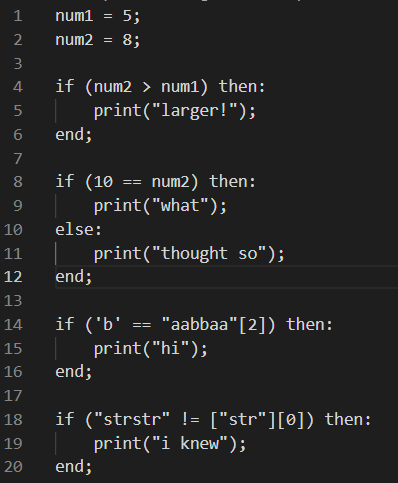
1. Функция соединения, join()
   1. Синтаксис: **join(strarray, str)**, где strarray – переменная или константа типа **strarray**, str – переменная или константа типа **string**
   2. Возвращаемое значение: константа типа **string**, представляющая из себя строку, полученную путём соединения всех строк strarray строкой str
   3. Примеры:



**Спецификация Straid: синтаксис управляющих конструкций**

Список управляющих конструкций:

1. Двухвариантный условный оператор, if-then-else
   1. Синтаксис:
      1. **if (condition) then: code\_block1;**
      2. **if (condition) then: code\_block1; else: code\_block2;**
   2. Примеры:



1. Итерирование строки (цикл по строке), iterate-track-as

2.1. Синтаксис

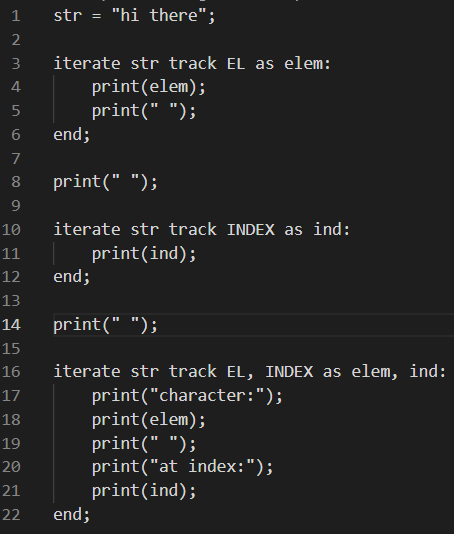
2.1.1. **iterate str track EL as имя\_переменной:**

2.1.2. **iterate str track INDEX as имя\_переменной:**

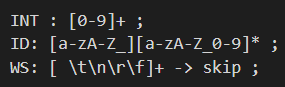
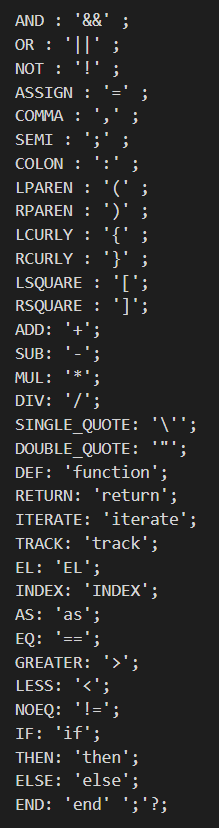
2.1.3. **iterate str track EL, INDEX as имя\_переменной1, имя\_переменной2:**

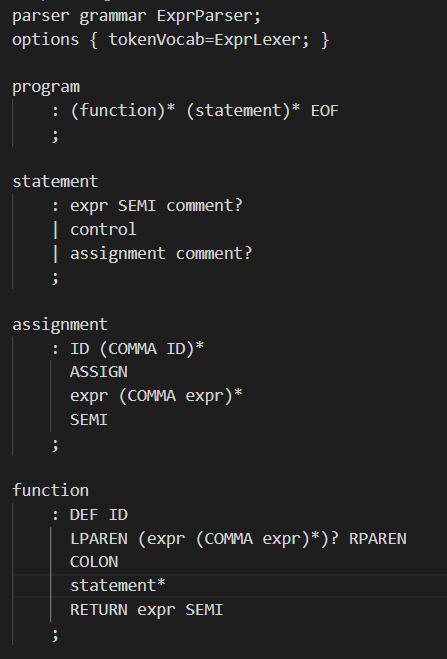
где str –переменная или константа типа **string**

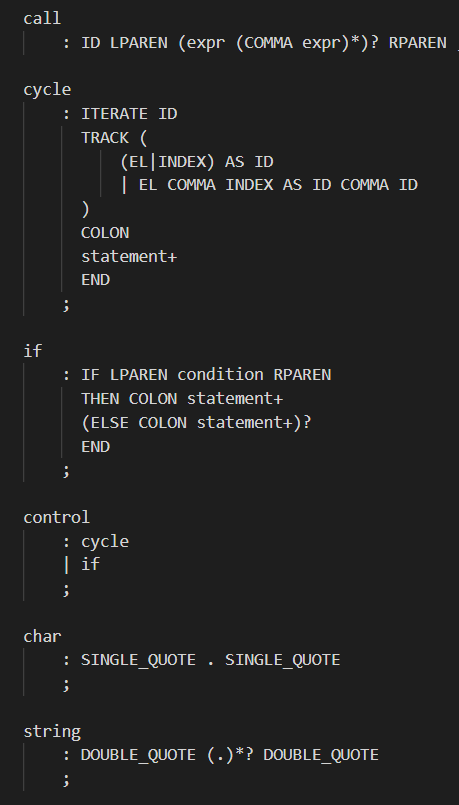
2.2. Примеры:

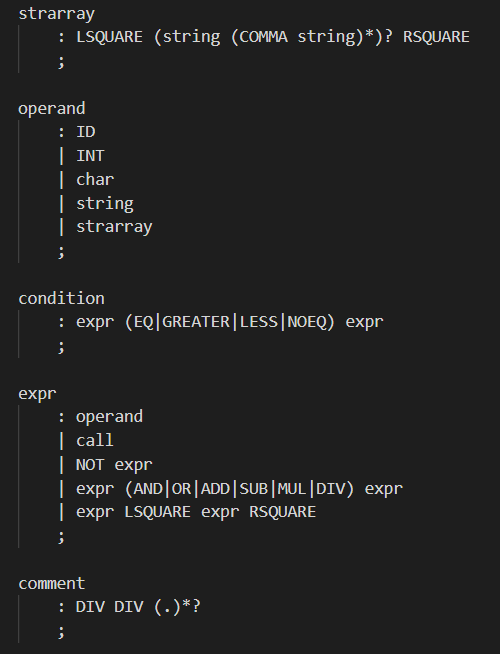


**Разработанная грамматика**









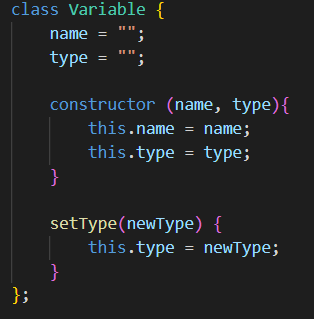
**Описание дополнительно разработанных классов**

Классы, разработанные для семантического анализатора:

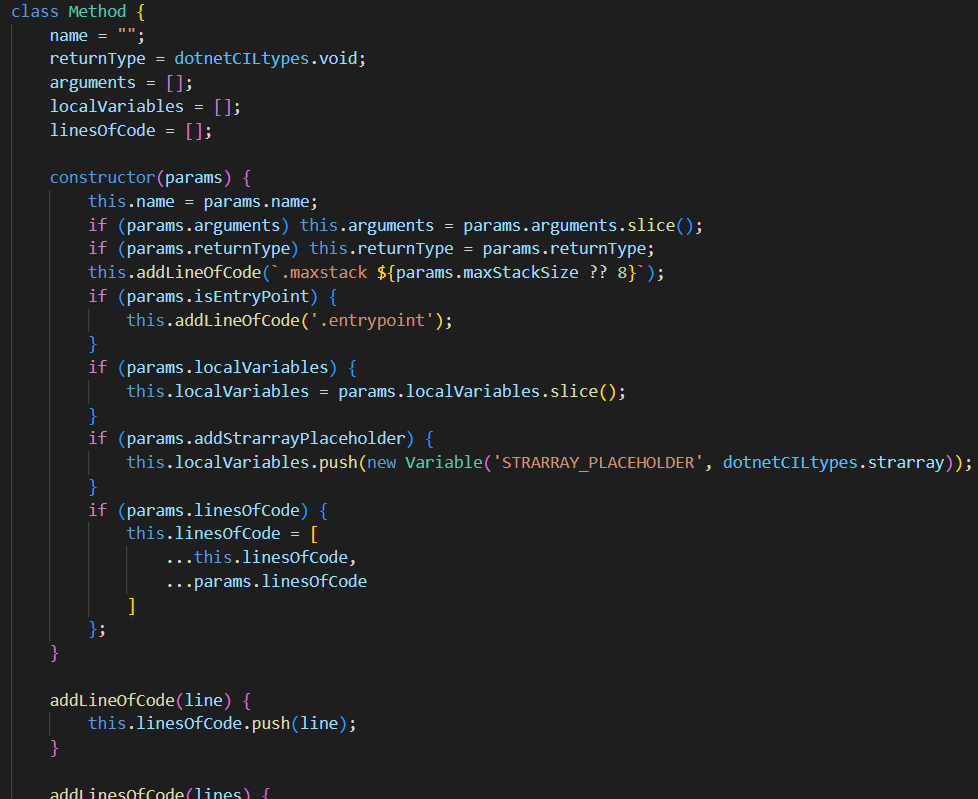
1. VariableDefinition – для учёта объявленных переменных и их типов
2. FunctionDefinition – для учёта объявленных пользовательских функций, их возвращаемых типов, типов аргументов и количества аргументов
3. Scope – для учёта областей видимости переменных

Классы, разработанные для компилятора:

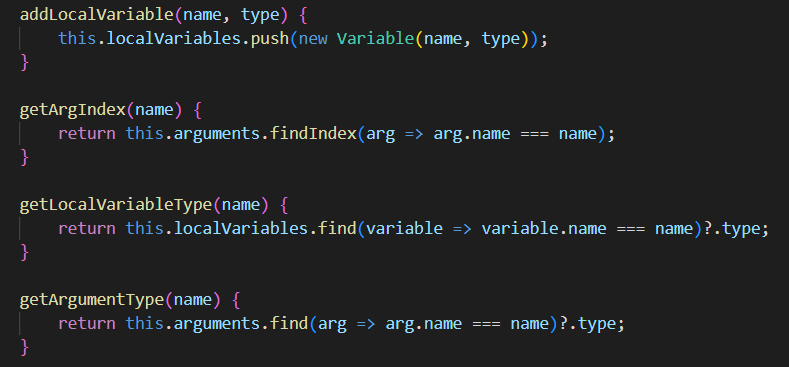
1. Variable – для учёта имён и типов объявленных переменных



1. Method – содержит полную информацию о методе, будь то пользовательский или предопределённый метод: имя, возвращаемое значение, аргументы (имена + типы), объявленные локальные переменные, строки кода метода

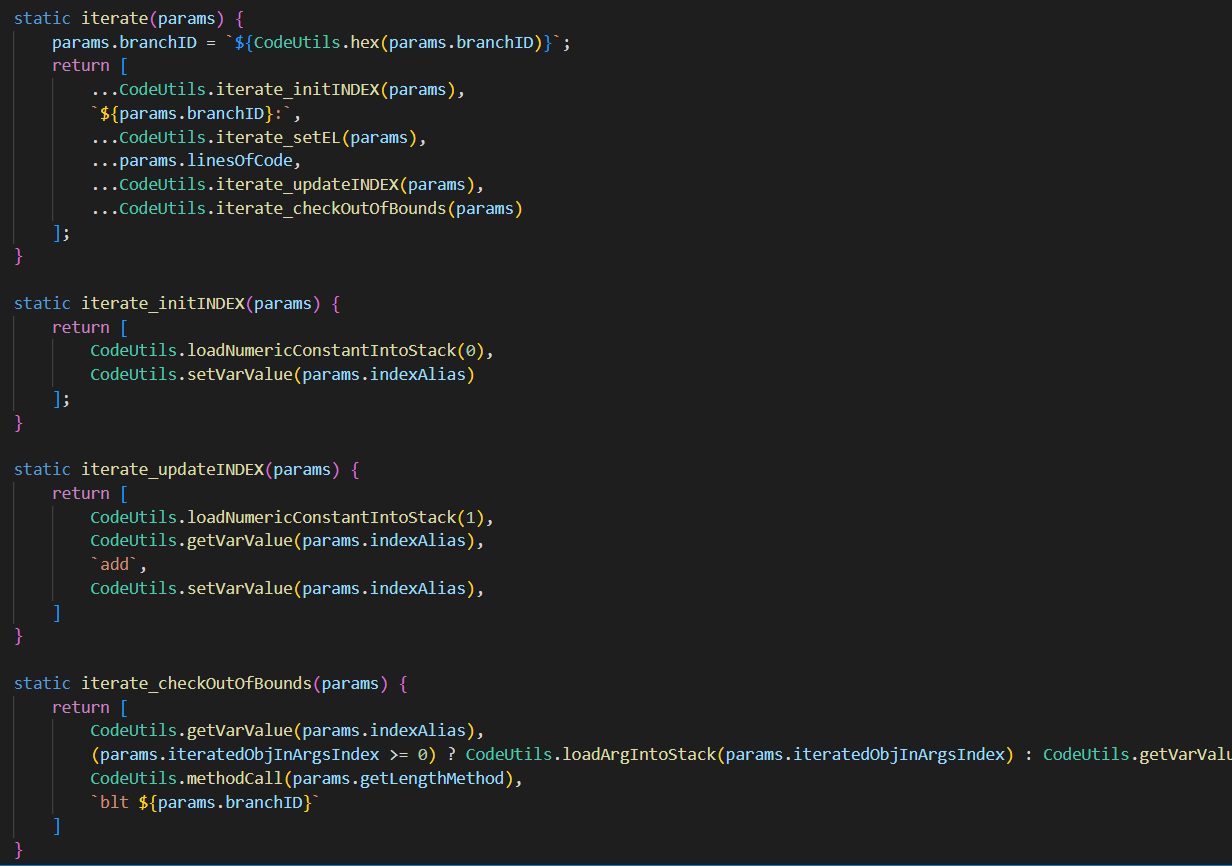


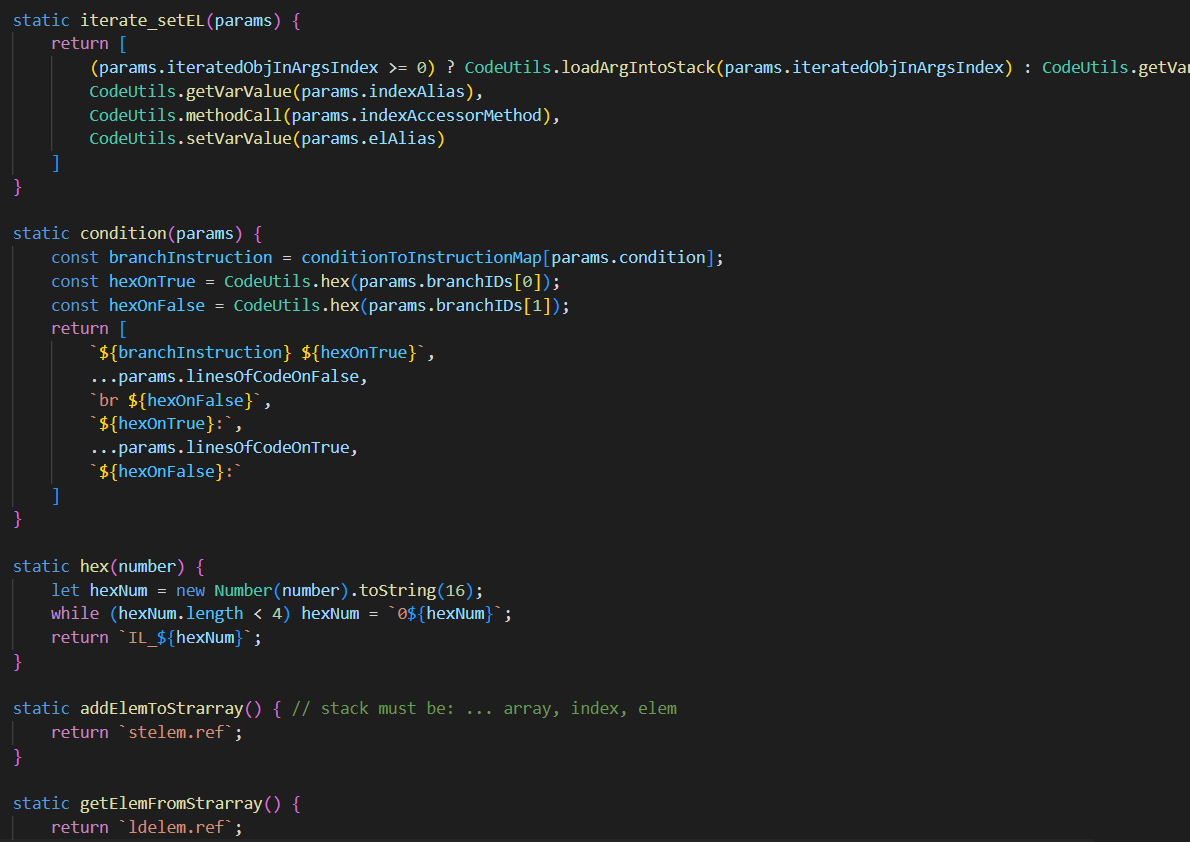




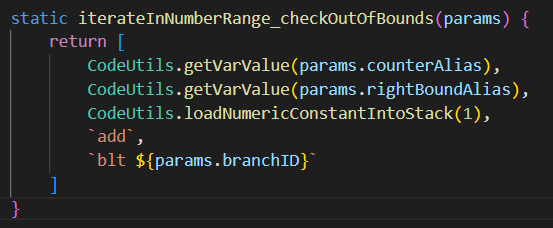
1. CodeUtils – вспомогательный класс, для более комфортного перевода кода в формате Straid в код .NET CIL







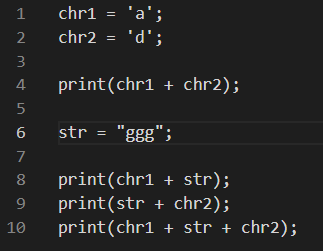




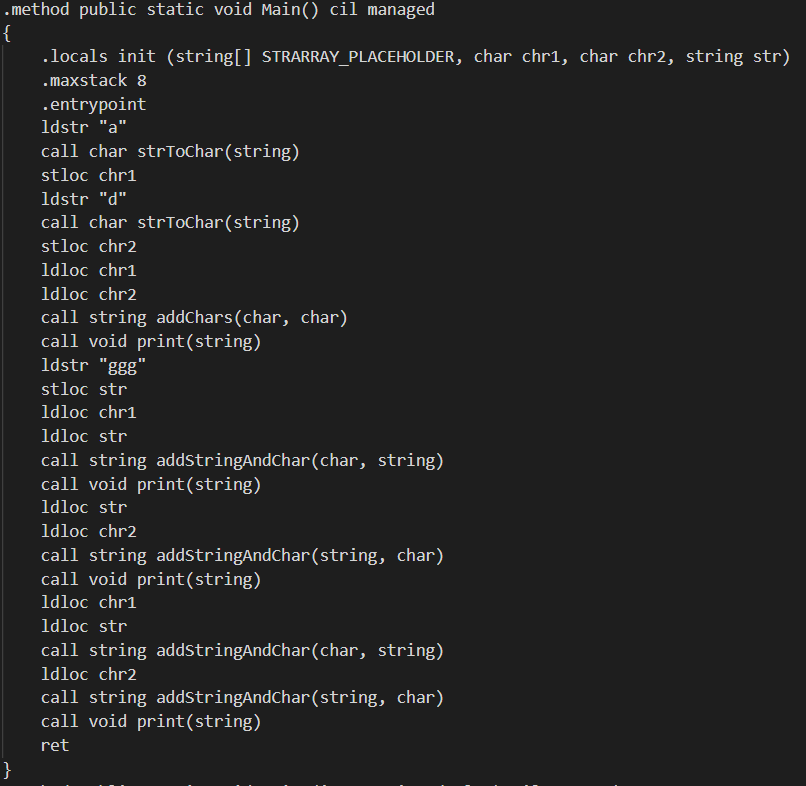
**Примеры работы компилятора**

Компилятор из Straid в .NET CIL перед началом непосредственно компиляции инициализирует предопределённые методы на языке .NET CIL, после чего компилирует непосредственно написанный пользователем код. В нижеследующих примерах запечатлена только та часть скомпилированного кода, которая соответствует исходному коду, написанному пользователем.

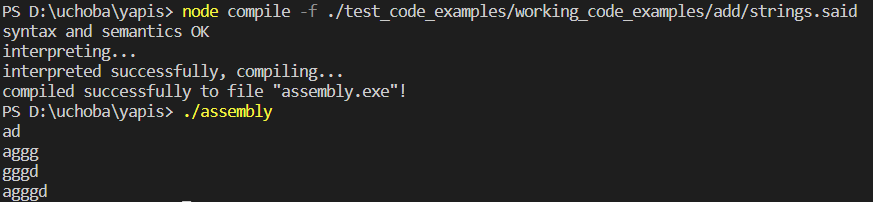
1. Пример 1
   1. Исходный код:



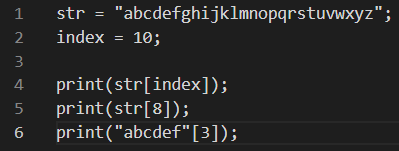
* 1. Результат компиляции:



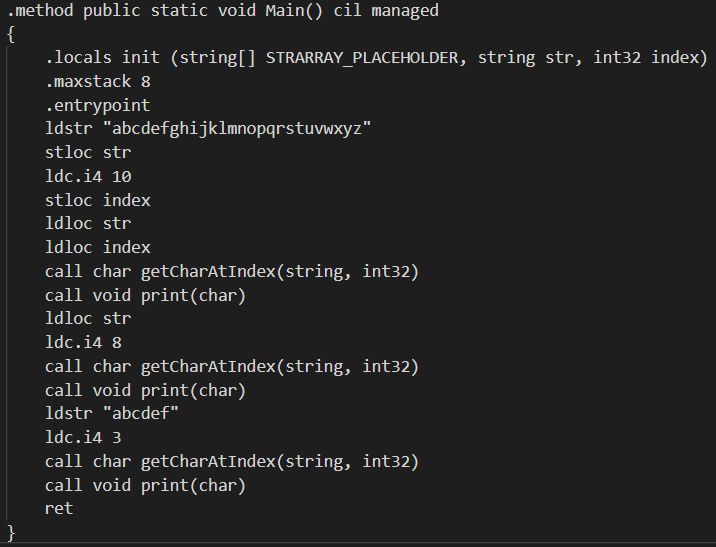
* 1. Консоль:



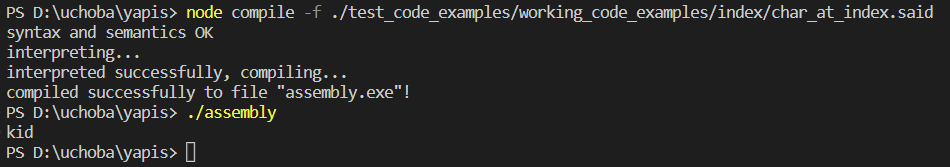
1. Пример 2
   1. Исходный код:



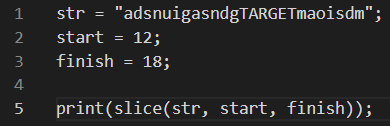
* 1. Результат компиляции:



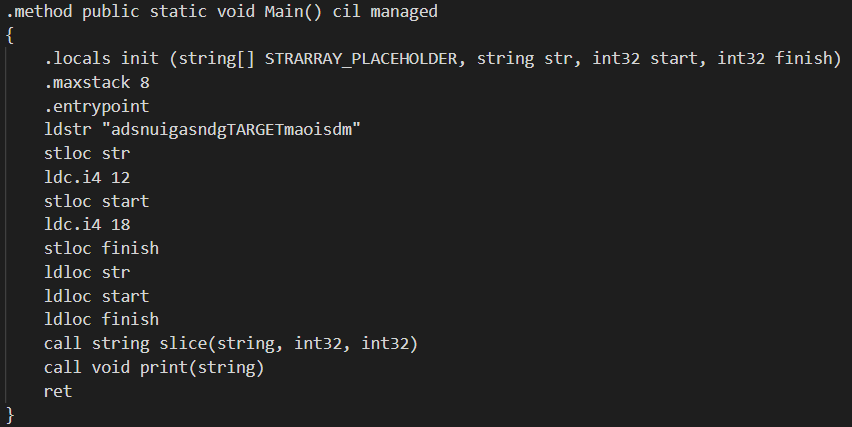
* 1. Консоль:



1. Пример 3
   1. Исходный код:



* 1. Результат компиляции:



* 1. Консоль:

