ДИСЦИПЛИНА **Конфигурационное управление**

(полное наименование дисциплины без сокращений)

ИНСТИТУТ **Информационных технологий**

КАФЕДРА **Корпоративных информационных систем**

(полное наименование кафедры)

ВИД УЧЕБНОГО МАТЕРИАЛА **Задание для текущего контроля**

(в соответствии с пп.1-11)

ПРЕПОДАВАТЕЛЬ **П.Н. Советов**

(фамилия, имя, отчество)

СЕМЕСТР **3 семестр (осенний) 2025/2026 учебного года**

(указать семестр обучения, учебный год)

Конфигурационное управление

Сборник домашних заданий

ИКБО-10-24

РТУ МИРЭА – 2025

Оглавление

О домашней работе

Домашняя работа (ДР) выполняется дистанционно. Результаты работы над ДР сохраняются в публично доступном git-репозитории. Ссылка на публично доступный git-репозиторий с результатами выполнения ДР загружается в СДО. Этапы работы над ДР должны быть отражены в истории коммитов с детальными сообщениями. Студент самостоятельно выбирает язык реализации и специализированный инструмент синтаксического разбора. Решения без использования специализированных инструментов синтаксического разбора не засчитываются.

Документация по ДР оформляется в виде readme.md, который содержит:

1. Общее описание.
2. Описание всех функций и настроек.
3. Описание команд для сборки проекта и запуска тестов.
4. Примеры использования.

Список публичных git-сервисов для репозиториев ДР:

1. github.com
2. gitea.com
3. gitlab.com
4. gitflic.ru
5. hub.mos.ru
6. gitverse.ru
7. gitee.com

Вариант №1

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
# Это однострочный комментарий
```

Многострочные комментарии:

```
=begin
Это многострочный
комментарий
=cut
```

Числа:

```
\d+
```

Словари:

```
{
  имя -> значение.
  имя -> значение.
  имя -> значение.
  ...
}
```

Имена:

```
[a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Словари.

Строки:

```
[[Это строка]]
```

Объявление константы на этапе трансляции:

```
имя is значение;
```

Вычисление константы на этапе трансляции:

```
$(имя)
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №2

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке yaml** попадает в стандартный вывод.

Числа:

```
-?(\d+|\d+\.\d*|\.\d+)([eE][-+]?\d+)?
```

Массивы:

```
array( значение, значение, значение, ... )
```

Словари:

```
{
  имя = значение;
  имя = значение;
  имя = значение;
  ...
}
```

Имена:

```
[a-zA-Z]+
```

Значения:

- Числа.
- Строки.
- Массивы.
- Словари.

Строки:

```
@"Это строка"
```

Объявление константы на этапе трансляции:

```
имя <- значение;
```

Вычисление константного выражения на этапе трансляции (постфиксная форма), пример:

```
^[имя 1 +]
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. `mod()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №3

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке json** попадает в стандартный вывод.

Однострочные комментарии:

```
|| Это однострочный комментарий
```

Числа:

```
0[bB][01]+
```

Массивы:

```
array( значение, значение, значение, ... )
```

Словари:

```
[
  имя => значение,
  имя => значение,
  имя => значение,
  ...
]
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
имя: значение;
```

Вычисление константы на этапе трансляции:

```
$имя$
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №4

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Числа:

```
\d*\.\d+
```

Массивы:

```
[ значение, значение, значение, ... ]
```

Словари:

```
{
  имя = значение
  имя = значение
  имя = значение
  ...
}
```

Имена:

```
[_a-zA-Z]+
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
def имя := значение
```

Вычисление константного выражения на этапе трансляции (постфиксная форма), пример:

```
.(имя 1 +).
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. Деление.
5. `min()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №5

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
* Это однострочный комментарий
```

Многострочные комментарии:

```
#|
Это многострочный
комментарий
|#
```

Числа:

```
\d+\.\d*
```

Словари:

```
[
  имя => значение,
  имя => значение,
  имя => значение,
  ...
]
```

Имена:

```
[_a-z]+
```

Значения:

- Числа.
- Строки.
- Словари.

Строки:

```
q(Это строка)
```

Объявление константы на этапе трансляции:

```
имя is значение;
```

Вычисление константы на этапе трансляции:

```
!{имя}
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №6

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
* Это однострочный комментарий
```

Числа:

```
[+-]?\d+\.\d+
```

Словари:

```
{
  имя => значение,
  имя => значение,
  имя => значение,
  ...
}
```

Имена:

```
[a-z][a-z0-9_]*
```

Значения:

- Числа.
- Строки.
- Словари.

Строки:

```
@"Это строка"
```

Объявление константы на этапе трансляции:

```
значение -> имя;
```

Вычисление константного выражения на этапе трансляции (инфиксная форма), пример:

```
^[имя + 1]
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. Деление.
5. `ord()`.
6. `print()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №7

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Многострочные комментарии:

```
{-
Это многострочный
комментарий
-}
```

Числа:

```
0[oO][0-7]+
```

Массивы:

```
[ значение, значение, значение, ... ]
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
@"Это строка"
```

Объявление константы на этапе трансляции:

```
имя := значение;
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
?(+ имя 1)
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. `ord()`.
4. `abs()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №8

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Числа:

```
\d+
```

Массивы:

```
#( значение, значение, значение, ... )
```

Словари:

```
{
  имя => значение,
  имя => значение,
  имя => значение,
  ...
}
```

Имена:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
значение -> имя
```

Вычисление константы на этапе трансляции:

```
[имя]
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №9

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Числа:

```
[+-]?\d+\.\d+
```

Массивы:

```
[ значение; значение; значение; ... ]
```

Имена:

```
[a-z][a-z0-9_]*
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
set имя = значение
```

Вычисление константы на этапе трансляции:

```
@{имя}
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №10

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
% Это однострочный комментарий
```

Многострочные комментарии:

```
<#
Это многострочный
комментарий
#>
```

Числа:

```
0[xX][0-9a-fA-F]+
```

Массивы:

```
[ значение, значение, значение, ... ]
```

Имена:

```
[a-z][a-z0-9_]*
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
@"Это строка"
```

Объявление константы на этапе трансляции:

```
(define имя значение)
```

Вычисление константного выражения на этапе трансляции (постфиксная форма), пример:

```
.{имя 1 +}.
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. `abs()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №11

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке yaml** попадает в стандартный вывод.

Однострочные комментарии:

```
# Это однострочный комментарий
```

Многострочные комментарии:

```
{
Это многострочный
комментарий
}
```

Числа:

```
0[bB][01]+
```

Массивы:

```
'( значение значение значение ... )
```

Имена:

```
[_a-zA-Z]+
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
var имя := значение
```

Вычисление константного выражения на этапе трансляции (постфиксная форма), пример:

```
|имя 1 +|
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. `pow()`.
4. `sqrt()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №12

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке toml** попадает в стандартный вывод.

Числа:

```
[+-]?\d+\.\d+
```

Словари:

```
table([
  имя = значение,
  имя = значение,
  имя = значение,
  ...
])
```

Имена:

```
[a-z][a-z0-9_]*
```

Значения:

- Числа.
- Словари.

Объявление константы на этапе трансляции:

```
значение -> имя;
```

Вычисление константы на этапе трансляции:

```
^{имя}
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №13

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке yaml** попадает в файл, путь к которому задан ключом командной строки.

Многострочные комментарии:

```
<#
Это многострочный
комментарий
#>
```

Числа:

```
\d+
```

Массивы:

```
'( значение значение значение ... )
```

Словари:

```
table([
  имя = значение,
  имя = значение,
  имя = значение,
  ...
])
```

Имена:

```
[a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
var имя значение;
```

Вычисление константы на этапе трансляции:

```
{имя}
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №14

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Числа:

```
[+-]?\d+
```

Словари:

```
begin
  имя := значение;
  имя := значение;
  имя := значение;
  ...
end
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Строки.
- Словари.

Строки:

```
q(Это строка)
```

Объявление константы на этапе трансляции:

```
имя is значение
```

Вычисление константного выражения на этапе трансляции (инфиксная форма), пример:

```
|имя + 1|
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. `ord()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №15

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
\ Это однострочный комментарий
```

Многострочные комментарии:

```
#[
Это многострочный
комментарий
]#
```

Числа:

```
\d+\.\d*
```

Словари:

```
{
  имя : значение,
  имя : значение,
  имя : значение,
  ...
}
```

Имена:

```
[a-zA-Z][a-zA-Z0-9]*
```

Значения:

- Числа.
- Словари.

Объявление константы на этапе трансляции:

```
global имя = значение
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
^(+ имя 1)
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. Деление.
5. `mod()`.
6. `abs()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №16

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке toml** попадает в стандартный вывод.

Многострочные комментарии:

```
(comment
Это многострочный
комментарий
)
```

Числа:

```
\d*\.\d+
```

Массивы:

```
{ значение. значение. значение. ... }
```

Имена:

```
[a-zA-Z]+
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
'Это строка'
```

Объявление константы на этапе трансляции:

```
global имя = значение;
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
.[+ имя 1].
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. `min()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №17

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
% Это однострочный комментарий
```

Числа:

```
0[bB][01]+
```

Массивы:

```
(list значение значение значение ... )
```

Словари:

```
table([
  имя = значение,
  имя = значение,
  имя = значение,
  ...
])
```

Имена:

```
[a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.
- Словари.

Строки:

```
'Это строка'
```

Объявление константы на этапе трансляции:

```
def имя := значение
```

Вычисление константы на этапе трансляции:

```
{имя}
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №18

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
:: Это однострочный комментарий
```

Числа:

```
[+-]?\d+\.?\d*[eE][+-]?\d+
```

Словари:

```
struct {
  имя = значение,
  имя = значение,
  имя = значение,
  ...
}
```

Имена:

```
[_A-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Словари.

Объявление константы на этапе трансляции:

```
имя = значение
```

Вычисление константы на этапе трансляции:

```
$(имя)
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №19

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке yaml** попадает в файл, путь к которому задан ключом командной строки.

Многострочные комментарии:

```
<!--
Это многострочный
комментарий
-->
```

Числа:

```
\d*\.\d+
```

Массивы:

```
[ значение значение значение ... ]
```

Имена:

```
[a-zA-Z][a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
'Это строка'
```

Объявление константы на этапе трансляции:

```
имя = значение
```

Вычисление константы на этапе трансляции:

```
.{имя}.
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №20

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
; Это однострочный комментарий
```

Числа:

```
0[bB][01]+
```

Массивы:

```
{ значение. значение. значение. ... }
```

Имена:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
[[Это строка]]
```

Объявление константы на этапе трансляции:

```
(define имя значение)
```

Вычисление константы на этапе трансляции:

```
#(имя)
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №21

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке yaml** попадает в стандартный вывод.

Многострочные комментарии:

```
=begin
Это многострочный
комментарий
=cut
```

Числа:

```
0[bB][01]+
```

Массивы:

```
{ значение. значение. значение. ... }
```

Имена:

```
[a-z][a-z0-9_]*
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
var имя = значение;
```

Вычисление константы на этапе трансляции:

```
[имя]
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №22

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке yaml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
! Это однострочный комментарий
```

Многострочные комментарии:

```
(*
Это многострочный
комментарий
*)
```

Числа:

```
[+-]?\d+\.?\d*[eE][+-]?\d+
```

Массивы:

```
[ значение; значение; значение; ... ]
```

Имена:

```
[a-zA-Z]+
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
global имя = значение;
```

Вычисление константного выражения на этапе трансляции (инфиксная форма), пример:

```
^(имя + 1)
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. `mod()`.
3. `sort()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №23

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке xml** попадает в стандартный вывод.

Однострочные комментарии:

```
NB. Это однострочный комментарий
```

Числа:

```
[+-]?\d+\.?\d*[eE][+-]?\d+
```

Массивы:

```
[ значение; значение; значение; ... ]
```

Имена:

```
[a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
q(Это строка)
```

Объявление константы на этапе трансляции:

```
имя: значение;
```

Вычисление константы на этапе трансляции:

```
#{имя}
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №24

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
! Это однострочный комментарий
```

Многострочные комментарии:

```
/*
Это многострочный
комментарий
*/
```

Числа:

```
[+-]?\d+\.?\d*[eE][+-]?\d+
```

Массивы:

```
{ значение, значение, значение, ... }
```

Словари:

```
$[
  имя : значение,
  имя : значение,
  имя : значение,
  ...
]
```

Имена:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.
- Словари.

Строки:

```
"Это строка"
```

Объявление константы на этапе трансляции:

```
имя <- значение;
```

Вычисление константного выражения на этапе трансляции (постфиксная форма), пример:

```
.(имя 1 +).
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. `sqrt()`.
5. `min()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №25

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке yaml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
C Это однострочный комментарий
```

Многострочные комментарии:

```
{#
Это многострочный
комментарий
#}
```

Числа:

```
0[xX][0-9a-fA-F]+
```

Массивы:

```
(list значение значение значение ... )
```

Имена:

```
[a-zA-Z][a-zA-Z0-9]*
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
имя: значение;
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
$[+ имя 1]
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. `sort()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №26

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
// Это однострочный комментарий
```

Числа:

```
0[oO][0-7]+
```

Массивы:

```
(list значение значение значение ... )
```

Словари:

```
{
  имя => значение,
  имя => значение,
  имя => значение,
  ...
}
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
имя := значение
```

Вычисление константы на этапе трансляции:

```
#[имя]
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №27

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке toml** попадает в стандартный вывод.

Однострочные комментарии:

```
-- Это однострочный комментарий
```

Многострочные комментарии:

```
=begin
Это многострочный
комментарий
=cut
```

Числа:

```
[+-]?\d+\.?\d*[eE][+-]?\d+
```

Массивы:

```
#( значение значение значение ... )
```

Словари:

```
begin
  имя := значение;
  имя := значение;
  имя := значение;
  ...
end
```

Имена:

```
[a-zA-Z][a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.
- Словари.

Строки:

```
'Это строка'
```

Объявление константы на этапе трансляции:

```
(define имя значение);
```

Вычисление константного выражения на этапе трансляции (инфиксная форма), пример:

```
![имя + 1]
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. `sort()`.
3. `max()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №28

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
! Это однострочный комментарий
```

Числа:

```
0[xX][0-9a-fA-F]+
```

Словари:

```
{
  имя => значение,
  имя => значение,
  имя => значение,
  ...
}
```

Имена:

```
[a-zA-Z]+
```

Значения:

- Числа.
- Словари.

Объявление константы на этапе трансляции:

```
var имя := значение;
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
#(+ имя 1)
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.

2. Вычитание.
3. Умножение.
4. `pow()`.
5. `sqrt()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №29

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
* Это однострочный комментарий
```

Числа:

```
[+-]?([1-9][0-9]*|0)
```

Массивы:

```
{ значение, значение, значение, ... }
```

Имена:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
var имя = значение
```

Вычисление константы на этапе трансляции:

```
$имя$
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №30

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
; Это однострочный комментарий
```

Числа:

```
\d*\.\d+
```

Массивы:

```
list( значение, значение, значение, ... )
```

Имена:

```
[a-zA-Z][a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
[[Это строка]]
```

Объявление константы на этапе трансляции:

```
var имя = значение
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
${+ имя 1}
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.

2. `min()`.
3. `abs()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №31

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
" Это однострочный комментарий
```

Числа:

```
[+-]?([1-9][0-9]*|0)
```

Массивы:

```
array( значение, значение, значение, ... )
```

Словари:

```
{
  имя : значение,
  имя : значение,
  имя : значение,
  ...
}
```

Имена:

```
[a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.
- Словари.

Строки:

```
[[Это строка]]
```

Объявление константы на этапе трансляции:

```
имя is значение
```

Вычисление константы на этапе трансляции:

```
$имя$
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №32

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке toml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
|| Это однострочный комментарий
```

Многострочные комментарии:

```
<!--
Это многострочный
комментарий
-->
```

Числа:

```
[1-9][0-9]*
```

Словари:

```
{
  имя : значение,
  имя : значение,
  имя : значение,
  ...
}
```

Имена:

```
[a-zA-Z]+
```

Значения:

- Числа.
- Словари.

Объявление константы на этапе трансляции:

```
var имя := значение;
```

Вычисление константного выражения на этапе трансляции (постфиксная форма), пример:

```
$(имя 1 +)
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. Деление.
5. `mod()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №33

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в стандартный вывод.

Однострочные комментарии:

```
REM Это однострочный комментарий
```

Числа:

```
0[oO][0-7]+
```

Словари:

```
struct {
  имя = значение,
  имя = значение,
  имя = значение,
  ...
}
```

Имена:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Словари.

Строки:

```
"Это строка"
```

Объявление константы на этапе трансляции:

```
var имя = значение;
```

Вычисление константы на этапе трансляции:

```
?[имя]
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №34

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке yaml** попадает в файл, путь к которому задан ключом командной строки.

Числа:

```
[+-]?\d+
```

Массивы:

```
({ значение, значение, значение, ... })
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
q(Это строка)
```

Объявление константы на этапе трансляции:

```
const имя = значение
```

Вычисление константы на этапе трансляции:

```
#[имя]
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №35

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке json** попадает в стандартный вывод.

Числа:

```
\d*\.\d+
```

Массивы:

```
#( значение значение значение ... )
```

Словари:

```
@{
  имя = значение;
  имя = значение;
  имя = значение;
  ...
}
```

Имена:

```
[_A-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.
- Словари.

Строки:

```
'Это строка'
```

Объявление константы на этапе трансляции:

```
(define имя значение)
```

Вычисление константы на этапе трансляции:

```
$[имя]
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №36

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке json** попадает в стандартный вывод.

Числа:

```
[+-]?\d+\.?\d*[eE][+-]?\d+
```

Массивы:

```
{ значение. значение. значение. ... }
```

Словари:

```
([
  имя : значение,
  имя : значение,
  имя : значение,
  ...
])
```

Имена:

```
[_a-zA-Z]+
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
имя <- значение;
```

Вычисление константы на этапе трансляции:

```
.(имя).
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №37

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке json** попадает в файл, путь к которому задан ключом командной строки.

Числа:

```
-?(\d+|\d+\.\d*|\.\d+)([eE][-+]?\d+)?
```

Массивы:

```
[ значение значение значение ... ]
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Массивы.

Объявление константы на этапе трансляции:

```
имя = значение
```

Вычисление константного выражения на этапе трансляции (префиксная форма), пример:

```
#{+ имя 1}
```

Результатом вычисления константного выражения является значение.

Для константных вычислений определены операции и функции:

1. Сложение.
2. Вычитание.
3. Умножение.
4. Деление.
5. `concat()`.
6. `sqrt()`.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №38

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
NB. Это однострочный комментарий
```

Многострочные комментарии:

```
{
Это многострочный
комментарий
}
```

Числа:

```
0[bB][01]+
```

Массивы:

```
[ значение, значение, значение, ... ]
```

Словари:

```
dict(
  имя = значение,
  имя = значение,
  имя = значение,
  ...
)
```

Имена:

```
[A-Z]+
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
def имя = значение;
```

Вычисление константы на этапе трансляции:

```
|имя|
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 2 примера описания конфигураций из разных предметных областей.

Вариант №39

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из стандартного ввода. Выходной текст на **языке xml** попадает в файл, путь к которому задан ключом командной строки.

Однострочные комментарии:

```
* Это однострочный комментарий
```

Многострочные комментарии:

```
{{!
Это многострочный
комментарий
}}
```

Числа:

```
[+-]?([1-9][0-9]*|0)
```

Массивы:

```
array( значение, значение, значение, ... )
```

Имена:

```
[a-zA-Z][a-zA-Z0-9]*
```

Значения:

- Числа.
- Строки.
- Массивы.

Строки:

```
@"Это строка"
```

Объявление константы на этапе трансляции:

```
var имя := значение;
```

Вычисление константы на этапе трансляции:

```
|имя|
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.

Вариант №40

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.

Входной текст на **учебном конфигурационном языке** принимается из файла, путь к которому задан ключом командной строки. Выходной текст на **языке xml** попадает в стандартный вывод.

Однострочные комментарии:

```
// Это однострочный комментарий
```

Числа:

```
0[xX][0-9a-fA-F]+
```

Массивы:

```
{ значение, значение, значение, ... }
```

Словари:

```
$[
  имя : значение,
  имя : значение,
  имя : значение,
  ...
]
```

Имена:

```
[a-zA-Z][_a-zA-Z0-9]*
```

Значения:

- Числа.
- Массивы.
- Словари.

Объявление константы на этапе трансляции:

```
значение -> имя
```

Вычисление константы на этапе трансляции:

```
|имя|
```

Результатом вычисления константного выражения является значение.

Все конструкции учебного конфигурационного языка (с учетом их возможной вложенности) должны быть покрыты тестами. Необходимо показать 3 примера описания конфигураций из разных предметных областей.