*Смольников Алексей ИКБО-10-21*

Задание 1

На jsonnet напишем программу, повторяющую заданный json. Для исключения повторения используется массив:

|  |
| --- |
| local arr = std.range(1, 24);  {  groups:["ИКБО-"+ i +"-20" for i in arr],  students: [  {age: 19, name: "Иванов Иван Иванович", group: "ИКБО-4-20"},  {age: 18, name: "Петров Петр Петрович", group: "ИКБО-5-20"},  {age: 19, name: "Сидоров Сидор Сидорович", group: "ИКБО-5-20"},  {age: 18, name: "Алексеев Алексей Алексеевич", group: "ИКБО-6-20"},  ],  subject: "Конфигурационное управление";  }  // {  // "groups": [  // "ИКБО-1-20",  // "ИКБО-2-20",  // "ИКБО-3-20",  // "ИКБО-4-20",  // "ИКБО-5-20",  // "ИКБО-6-20",  // "ИКБО-7-20",  // "ИКБО-8-20",  // "ИКБО-9-20",  // "ИКБО-10-20",  // "ИКБО-11-20",  // "ИКБО-12-20",  // "ИКБО-13-20",  // "ИКБО-14-20",  // "ИКБО-15-20",  // "ИКБО-16-20",  // "ИКБО-17-20",  // "ИКБО-18-20",  // "ИКБО-19-20",  // "ИКБО-20-20",  // "ИКБО-21-20",  // "ИКБО-22-20",  // "ИКБО-23-20",  // "ИКБО-24-20"  // ],  // "students": [  // {  // "age": 19,  // "group": "ИКБО-4-20",  // "name": "Иванов Иван Иванович"  // },  // {  // "age": 18,  // "group": "ИКБО-5-20",  // "name": "Петров Петр Петрович"  // },  // {  // "age": 19,  // "group": "ИКБО-5-20",  // "name": "Сидоров Сидор Сидорович"  // },  // {  // "age": 18,  // "group": "ИКБО-6-20",  // "name": "Алексеев Алексей Алексеевич"  // }  // ],  // "subject": "Конфигурационное управление"  // } |

Задание 2

Реализуем аналогичное задание на Dhall

|  |
| --- |
| let students = [{age = 19, name= "Иванов И И", group = "ИКБО-4-21"},{age = 18, name= "Петров П П", group = "ИКБО-4-20"}, {age = 18, name= "Сидоров С С", group = "ИКБО-4-20"}, {age = 18, name = "Алексеев Алексей Алексеевич", group = "ИКБО-6-20"},]  let groups = ["ИКБО-1-20",  "ИКБО-2-20",  "ИКБО-3-20",  "ИКБО-4-20",  "ИКБО-5-20",  "ИКБО-6-20",  "ИКБО-7-20",  "ИКБО-8-20",  "ИКБО-9-20",  "ИКБО-10-20",  "ИКБО-11-20",  "ИКБО-12-20",  "ИКБО-13-20",  "ИКБО-14-20",  "ИКБО-15-20",  "ИКБО-16-20",  "ИКБО-17-20",  "ИКБО-18-20",  "ИКБО-19-20",  "ИКБО-20-20",  "ИКБО-21-20",  "ИКБО-22-20",  "ИКБО-23-20",  "ИКБО-24-20"]  in {  groups,  students,  subject= "Конфигурационное управление"  }  -- {  -- "groups": [  -- "ИКБО-1-20",  -- "ИКБО-2-20",  -- "ИКБО-3-20",  -- "ИКБО-4-20",  -- "ИКБО-5-20",  -- "ИКБО-6-20",  -- "ИКБО-7-20",  -- "ИКБО-8-20",  -- "ИКБО-9-20",  -- "ИКБО-10-20",  -- "ИКБО-11-20",  -- "ИКБО-12-20",  -- "ИКБО-13-20",  -- "ИКБО-14-20",  -- "ИКБО-15-20",  -- "ИКБО-16-20",  -- "ИКБО-17-20",  -- "ИКБО-18-20",  -- "ИКБО-19-20",  -- "ИКБО-20-20",  -- "ИКБО-21-20",  -- "ИКБО-22-20",  -- "ИКБО-23-20",  -- "ИКБО-24-20"  -- ],  -- "students": [  -- {  -- "age": 19,  -- "group": "ИКБО-4-21",  -- "name": "Иванов И И"  -- },  -- {  -- "age": 18,  -- "group": "ИКБО-4-20",  -- "name": "Петров П П"  -- },  -- {  -- "age": 18,  -- "group": "ИКБО-4-20",  -- "name": "Сидоров С С"  -- },  -- {  -- "age": 18,  -- "group": "ИКБО-6-20",  -- "name": "Алексеев Алексей Алексеевич"  -- }  -- ],  -- "subject": "Конфигурационное управление"  -- } |

Задание 3

Для имплементации синтаксического анализатора воспользуемся простейшими компонентами языка Python, итеративно перебирая весь файл, если попадется нежелательный элемент, возвращается false:

|  |
| --- |
| import os  import sys  # read arguments given to program  def read\_args():  args = []  for arg in sys.argv[1:]:  args.append(arg)  return args  def checkline(line):  return all(c in '01\n' for c in line)  def main(args):  if args[0] == '-h':  print('Usage: binAnalyzer.py <path>')  return  if len(args)==0:  print('Usage: binAnalyzer.py <path>')  return    file = open(args[0], 'r')      if all(checkline(line) for line in file.readlines()):  print('Success')  else:  print('Failure')    return 0  if \_\_name\_\_ == '\_\_main\_\_':  main(read\_args()) |

Задание 4.

По заданному набору данных имплементируем синтаксический анализатор скобочного языка. Видно, что каждое слово должно быть полиномом, состоящим из определенных символов:

|  |
| --- |
| import os  import sys  # read arguments given to program  def read\_args():  args = []  for arg in sys.argv[1:]:  args.append(arg)  return args  def checkLine(line):  lenl = len(line)    if line[-1] == '\n':  line = line[:-1]  lenl -= 1    if lenl%2 != 0 or not all(c in '(){}\n' for c in line):  return False    for i in range(0, lenl//2):  if line[i] == '(' and line[lenl-i-1] != ')':  return False  if line[i] == '{' and line[lenl-i-1] != '}':  return False    return True        def main(args):  if len(args)==0 or args[0] == '-h' :  print('Usage: bracketsAnalyzer.py <path>')  return    if all(checkLine(line) for line in open(args[0], 'r').readlines()):  print('Success')  else:  print('Failure')    return 0  if \_\_name\_\_ == '\_\_main\_\_':  main(read\_args()) |