Запуск программы

Программа запускается через терминал тремя способами. Первый и самый простой, следующую команду надо вбить в терминал в папке где находится программа:

```
> aivika-model
```

Программа запустится и попытается прочитать входные данные из файла `input.txt` и записать выходные данные в `output.txt`.

Второй способ:

```
> aivika-model myinput.txt
```

Отличие в том, что программа прочтет данные из файла `myinput.txt`, а не из файла по умолчанию.

Третий способ:

```
> aivika-model myinput.txt myoutput.txt
```

Отличие в том, что программа прочтет данные из файла `myinput.txt` и запишет данные в файл `myoutput.txt`.

> **Замечание**: Во время работы программа выводит информацию в терминал, для Windows вместо русских букв будут выведены нечитаемые символы (так как терминал Windows не поддерживает кодировку UTF8). Пользователь не должен нервничать, так как весь вывод дублируется в файле `output.txt` или указанном при запуске.

Формат входных данных

Программа использует свой собственный человекочитаемый формат для входных данных. Рассмотрим пример входных данных:

```
experiment {
  input erlang 10 2
  subsystem {
    processing exponent 10
    buffer 3
  }
  subsystem {
    processing normal 10 2
    buffer 3
  }
  simulation time 500000
  output precision 5
}
```

Первая строчка объявляет, что далее следует блок эксперимента. Блоков экспериментов может быть несколько, каждый такой блок кодирует отдельный запуск симуляции и сбор статистики для нее. В данном примере объявлен только один блок эксперимента.

Вторая строчка `input erlang 10 2` сообщает программе, что входной поток системы будет распределен по Эрлангу со средним временем `t = 10` и `k = 2`. Поддерживаются следующие распределения:

- `erlang t k` - Эрланговсоке распределение со средним временем `t` (число с плавающей запятой) и коэффициентом `k` (натуральное число). Например: `erlang 20.5 4`
- `exponent t` - Экспоненциальное распределение со средним временем `t` (число с плавающей запятой). Например: `exponent 14.2`
- `normal mu sigma` - Нормальное распределение с математическим ожиданием `mu` (число с плавающей запятой) и среднеквадратическим отклонением `sigma` (число с плавающей запятой). Например: `normal 10.1 2.5`

- `uniform minVal maxVal` - Равномерное распределение с минимальным значением `minVal` (число с плавающей запятой) и максимальным значением `maxVal` (число с плавающей запятой).
- `hyperexponent (t1, p1) .. (tn, pn)` - Гипер-экспоненциальное распределение со списком параметров, состоящих из пар: среднее время `tn` и вероятность `pn`. Пар параметров может быть неограниченное число. Сумма всех вероятностей должна давать `1.0`. Например: `hyperexponent (10.5, 0.1) (5.0, 0.8) (3.2, 0.1)`

После определения входного потока идет список объявления подсистем, пример одной подсистемы:

```
subsystem {
  processing exponent 10
  buffer 3
}
```

Число подсистем неограничено сверху. С помощью `processing exponent 10` задается распределение времени обслуживания операционного автомата в данной подсистеме. Формар задания распределений аналогичен формату входного потока. Другой пример распределения обслуживания: `processing erlang 10.5 2`.

Далее следует задание размера буфера очереди подсистемы: `buffer 3`. Размер буфера должен быть натуральным числом.

После всех подсистем идет задание длительности иммитационного моделирования: `simulation time 500000`. Чем больше это число, тем дольше будет выполняться программа и тем точнее будет результат (при очень больших значениях программа падает по неизвестным разработчикам причинам).

Последняя строка `output precision 5` задает количество знаков после запятой при выводе результатов. Должно быть целым положительным числом.