Задача 6. Чтение JSON

Источник:	повышенной сложности I
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.bin`
Ограничение по времени:	разумное
Ограничение по памяти:	разумное

В этой задаче нужно прочитать JSON с помощью библиотеки json-c и записать его в формате BSON.

Спецификация формата JSON есть на сайте json.org. Любое *значение* в формате JSON имеет один из следующих типов:

- Строка — произвольная последовательность символов Unicode, задаётся в кодировке UTF-8 и заключается в двойные кавычки
- Число — в общем случае вещественное с двойной точностью
- Булево значение `true` или `false`
- Нулевое значение `null`
- Массив — элементы могут быть любыми значениями: они разделяются запятыми, всё содержимое массива заключается в квадратные скобки
- Объект — список пар вида «ключ: значение», где ключ — это строка, а значение может быть любым значением

Во входном файле записано ровно одно значение, которое является либо объектом, либо массивом. Формат записи таков, что библиотека `json-c` нормально его читает с настройками по умолчанию.

Формат BSON является бинарным аналогом формата JSON, его спецификация есть на сайте bsonspec.org. Она может быть проще для понимания, чем идущее далее описание. В данной задаче никакая строка не будет содержать нулевой байт — в этом случае любое JSON-значение можно записать в формате BSON как последовательность байтов.

При конвертации в BSON каждое поле объекта и каждый элемент массива записывается в следующем виде: сначала один байт, кодирующий тип значения, потом имя поля, и наконец его значение. У элемента массива имя поля определяется как его номер в массиве, записанный в десятичной системе исчисления.

Следует использовать следующие типы BSON:

- Строка — тип 2
- Число — тип 1
- Булево значение — тип 8
- Нулевое значение `null` — тип 10
- Массив — тип 4
- Объект — тип 3

Значение типа «строка» записывается так: сначала 32-битное целое число, равное количеству байт в строке плюс один, затем сама строка как последовательность байтов в кодировке UTF-8, и наконец нулевой байт. Значение типа «число» записывается в 8 байт как значение типа `double`. Булево значение записывается одним байтом: 0 для `false` и 1 для `true`. Значение `null` записывается пустой последовательностью байтов. Массив или объект записывается так: сначала 32-битное целое, равное количеству байтов в записи объекта/массива, затем все элементы массива или поля массива в нужном порядке, и наконец нулевой байт.

Имя поля записывается почти как строка, но длина в начале не записывается. То есть пишется сразу содержимое строки, и в конце нулевой байт.

Что рекомендуется посмотреть по библиотеке `json-c` в первую очередь:

1. Пример парсинга test2.c.
2. Хедер json_visit.h.
3. Хедер json_object.h.

В систему тестирования нужно посылать один файл с исходным кодом, в котором можно подключать хедеры библиотеки `json-c` (без путей). При сборке ваш файл будет компилироваться и линковаться со статической библиотекой `json_c.lib`.

Пример

input.txt

```
{
  "array": [
    1,
    2,
    3
  ],
  "boolean": true,
  "color": "#82b92c",
  "null": null,
  "number": 123,
  "object": {
    "a": "b",
    "c": "d",
    "e": [ 5, {"key": -9} ]
  },
  "my string": "Hello World",
  "codename": "3/6_json"
}
```

output.bin

```
DE 00 00 00 04 61 72 72 61 79 00 26 00 00 00 01 30 00 00 00 00 00 00 00
F0 3F 01 31 00 00 00 00 00 00 00 00 40 01 32 00 00 00 00 00 00 00 08 40
00 08 62 6F 6F 6C 65 61 6E 00 01 02 63 6F 6C 6F 72 00 08 00 00 00 23 38
32 62 39 32 63 00 0A 6E 75 6C 6C 00 01 6E 75 6D 62 65 72 00 00 00 00 00
00 C0 5E 40 03 6F 62 6A 65 63 74 00 3F 00 00 00 02 61 00 02 00 00 00 62
00 02 63 00 02 00 00 00 64 00 04 65 00 25 00 00 00 01 30 00 00 00 00 00
00 00 14 40 03 31 00 12 00 00 00 01 6B 65 79 00 00 00 00 00 00 00 22 C0
00 00 00 02 6D 79 20 73 74 72 69 6E 67 00 0C 00 00 00 48 65 6C 6C 6F 20
57 6F 72 6C 64 00 02 63 6F 64 65 6E 61 6D 65 00 09 00 00 00 33 2F 36 5F
6A 73 6F 6E 00 00
```

Комментарий

Пример в нормальном виде можно скачать отсюда.

Хедеры `config.h` и `json_config.h` можно скачать отсюда. С ними можно собрать статическую библиотеку `json-c` напрямую компилятором. Без них придётся запускать CMake. В системе тестирования эти хедеры будут присутствовать, как и все другие хедеры библиотеки.