2 Алгоритмы I

Написать программу, которая выполняет следующие действия:

1. Заполняет `std::vector< DataStruct >` структурами `DataStruct`, прочитанными со стандартного ввода. Чтение необходимо осуществлять с помощью итераторов и алгоритмов STL (`std::copy`, итераторы потока и перегрузки оператора побитового сдвига для чтения из потока)
2. Сортирует считанные данные следующим образом:
   (a) По возрастанию `key1`
   (b) По возрастанию `key2`, если `key1` одинаковые
   (c) По возрастанию длины строки `key3`, если прочие поля равны
3. Выводит результаты сортировки на стандартный вывод. Каждая строка должна содержать ровно один объект. Формат каждого выводимого объекта аналогичен формату ввода. Вывод необходимо осуществлять с помощью итераторов и алгоритмов STL (`std::copy`, итераторов потока и перегрузки оператора побитового сдвига для вывода в поток)

Входные данные могут содержать строки с неподдерживаемым форматом данных. Такие строки должны игнорироваться. Формат данных, который необходимо обрабатывать зависит в том числе от определения `DataStruct`

1. Тип `DataStruct` определён следующим образом:

```
struct DataStruct
{
  /* TYPE1 */ key1;
  /* TYPE2 */ key2;
  std::string key3;
};
```

   Конкретные типы полей `key1` и `key2`, а также соответствующий полям формат должны быть указаны преподавателем

2. Каждая запись ограничена парой скобок. Внутри этих скобок в качестве разделителей используются пробельные символы и символы `:`. Например:

```
(:key1 10ull:key2 'c':key3 "Data":)
```

   Порядок описания полей в структуре не определён. Например следующие структуры данных считаются идентичными:

```
(:key1 10ull:key2 'c':key3 "Data":)
(:key2 'c':key1 10ull:key3 "Data":)
(:key3 "Data":key2 'c':key1 10ull:)
```

   Имя поля и соответствующее значение гарантировано разделены ровно одним пробелом. Символы двоеточия гарантировано примыкают к прочим элементам записи. При выводе в поток поля выводятся в том порядке, в котором они заданы в структуре (запоминать исходный порядок не требуется)

3. Поля могут иметь следующий тип и соответствующий формат:

   - `[DBL LIT]` Вещественное поле с двойной точностью (**`double`**) в формате литерала:

     ```
     :keyX 50.0d:
     ```

   - `[DBL SCI]` Вещественное поле с двойной точностью (**`double`**) в научном формате:

     ```
     :keyX 5.45e-2:
     :keyX 5.45E-2:
     ```

   - `[SLL LIT]` Знаковое максимально доступной ёмкости (**`long long`**) в формате литерала:

     ```
     :keyX 89ll:
     :keyX -89LL:
     ```

- `[ULL LIT]` Беззнаковое максимально доступной ёмкости (**unsigned long long**) в формате литерала:

```
:keyX 89ull:
:keyX 89ULL:
```

- `[ULL OCT]` Беззнаковое максимально доступной ёмкости (**unsigned long long**) в формате восьмиричного литерала:

```
:keyX 076:
:keyX 01001:
```

- `[ULL BIN]` Беззнаковое максимально доступной ёмкости (**unsigned long long**) в формате двоичного литерала:

```
:keyX 0b1000101:
:keyX 0b001001:
```

- `[ULL HEX]` Беззнаковое максимально доступной ёмкости (**unsigned long long**) в формате шестнадцатиричного литерала:

```
:keyX 0xFFFA:
:keyX 0x0100F:
```

- `[CHR LIT]` Символ (**char**) в формате символьного литерала:

```
:keyX 'c':
:keyX 'a':
```

- `[CMP LSP]` Комплексное число (`std::complex< double >`) в следующем виде:

```
:keyX #c(1.0 -1.0):
:keyX #c(-1.0 1.0):
```

  Гарантируется, что вещественная и мнимая часть разделены ровно одним пробелом. При сравнении с другими полями должен быть использован модуль комплексного числа

- `[RAT LSP]` Рациональное число (`std::pair< long long, unsigned long long >`) в следующем виде:

```
:keyX (:N -2:D 3:):
:keyX (:N 3:D 2:):
```

- Если в качестве варианта задания выданы `[RAT LSP]` и `[CMP LSP]`, то `DataStruct` должна быть определена следующем образом:

```
struct DataStruct
{
  std::pair< long long, unsigned long long > key1;
  std::complex< double > key2;
  std::string key3;
};
```

  При этом обрабатываемые записи имеют следующий вид:

```
(:key1 #c(1.0 -1.0):key2 (:N -1:D 5:):key3 "data":)
(:key2 (:N -1:D 5:):key3 "with : inside":key1 #c(2.0 -3.0):)
```