# Задание по STL

Реализовать шаблонный класс

`template <typename T1, typename T2> Bijection`

взаимно-однозначного отображения (биекции) объектов типа `T1` в объекты типа `T2`.

Реализация должна быть корректной для любых аргументов `T1`, `T2`, для которых определены

- конструктор по умолчанию: `T1()`, `T2()`,
- конструктор копирования: `T1(const T1 &)`, `T2(const T2 &)`,
- оператор строгого сравнения:
  `bool operator <(const T1 &, const T1 &)`, `bool operator <(const T2 &, const T2 &)`.

Объекты `x`, `y` считаются *равными* (то есть одним и тем же элементом), если `!((x < y) && (y < x))`.
**Интерфейс** класса:

- `Bijection()`: **инициализация** отображения с пустой областью определения.
- `void add(const T1 & x, const T2 & y)`: **добавление образа**;
  - пусть $f'$ — отображение, отличающееся от текущего только тем, что $f(x) = y$:
    - если область определения $f$ не содержит элемента, равного $x$, то $f'$ — расширенное отображение,
    - иначе $f'$ — отображение, отличающееся от $f$ заменой образа элемента $x$ на $y$;
  - если $f'$ — биекция, то в результате вызова метода отображение $f$ преобразуется в $f'$;
  - иначе реализуемое отображение $f$ не изменяется.
- `Bijection<T2,T1> reverse() const`: **обращение**; вызовом возвращается отображение, обратное к текущему.

Класс должен быть совместим с циклом по коллекции (range-based for), в котором перебираются константные элементы области значений (`const T2 &`) по возрастанию их прообразов. В **рекомендуемом способе** класс биекции содержит:

- класс итератора `Bijection<T1, T2>::Iterator`;
  - для итераторов опередлны операторы `++` (префиксный), `==`, `*` (унарный, разыменование);
- метод `Bijection<T1, T2>::Iterator begin() const`, возвращающий итератор, указывающий на начало перебора;
- метод `Bijection<T1, T2>::Iterator end() const`, возвращающий итератор, указывающий на конец перебора;

**Идеальная сложность** методов интерфейса по времени работы:

- инициализация: $O(1)$;
- добавление образа: амортизированная $O(log\ n)$;
- обращение: $O(N)$, где $N$ — размер текущей области определения биекции;
- операторы и методы, относящиеся к итераторам: амортизированная $O(1)$.

В оценках сложности полагается, что копирование объектов типов `T1`, `T2` производится за время $O(1)$.
**Ограничения:**

- Запрещено использование “чистых указателей” (типа `T *`) и нестандартных библиотек.
  - То есть по умолчанию всё пишется в STL.
- Запрещено хранить в классе биекции более одной копии каждого элемента — как области определения, так и области значений.

**Критерии оценки:**

- Корректно работающая реализация с интерфейсом идеальной сложности: 100 баллов.
- Некорректно работающая реализация: штрафы в зависимости от “тяжести” ошибок.
- Неидеальная сложность: штрафы в зависимости от того, насколько полученная сложность хуже идеальной.