# Постановка задачи лабораторной №1 Множества

## Условия и ограничения

- Рассматриваются только конечные множества
- Размер множества не превышает 2^32 – 1
- Элементы множества не хранятся

## Пример

Допустим, максимальный размер множества 10:

Множество U (универс) `{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }`

Подмножество универса множество A `{ 1, 5, 7, 8, 10 }`

Характеристический вектор для A `{ 1, 0, 0, 0, 1, 0, 1, 1, 0,  1  }`

## Реализация через битовые строки

Эффективный способ реализации множеств – битовое поле (битовая строка)

Битовые поля также уже были реализованы:

- `std::vector<bool>` – для динамически изменяемых битовых полей
- `std::bitset` – для битовых полей с неизменяемым размером (подходит данной задаче больше, так как размер универса зафиксирован, отсюда можно позаимствовать идеи для методов)

Мы же спроектируем наш собственный класс `TBitField`.

## Тонкости реализации

1. `elem_type` - любой беззнаковый целочисленный тип (придётся использовать маски битов) или структуры с битовыми полями

2. ПРО `bitLen` и `memLen`

- если `bitLen = 30`, размер `memLen` равен 1 (если используется 32-битный или больше размер `elem_type`)
- если `bitLen = 30`, размер `memLen` равен 3 (если используется, например, 8-битный размер elem_type)

**Пример.** Множество `A = { 1, 5, 7, 8, 9 }`, максимально возможное значение 10.

| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | | | | | | | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| unsigned char (209) | | | | | | | | unsigned char (1) | | | | | | | |

Здесь: `bitLen = 12, memLen = 2.`

Запомни! Биты в числе расположены слева направо!

3. Множества можно реализовать через **наследование** или **агрегацию**. Публичное наследование наследует интерфейс битовых полей, который не согласуется с представлением математического объекта Множество. Лучше агрегация. Однако есть еще private-наследование (по факту агрегация с дополнительными удобными свойствами).

4. Обращение к битам и применение побитовых операций требуют реализации **масок.**

Маски позволяют узнать индекс в массиве pMem, то есть в каком байте (элементе типа unsigned char) находится бит отвечающий за элемент множества, а также позицию данного элемента внутри данного бита. Пояснения к служебным полям:

```
// количество бит, которое необходимо для хранения данного типа в памяти
// (у нас в примере 8 бит для 1-байтового unsigned char)
size_t bitsInElem = std::numeric_limits<elem_type>::digits;

// длина битового представления для заданного числа
// проще говоря нужно определить степень двойки, чтобы делать сдвиги битовые
// 2^x = (8 - 1) -> x = 3
size_t shiftSize = std::bit_width(bitsInElem - 1);
```

**Пояснения к битовым операциям:**

- установить бит


- очистить бит


- узнать значение бита

## Обязательные требования к реализации

1. Использовать модули по аналогии с лекциями – советую пересмотреть видеоматериалы (C++20). Экспортировать класс Битовые поля.
2. Подключение библиотек в соответствии с новым стандартом:

```
import std.core;

using std::ostream;
```

3. Использовать исключения

```
using std::out_of_range;

using std::lenght_error;
```

4. Оператор присваивания реализовать не как обычно, а через move-конструктор и `swap`.
5. Использовать `const` (для методов, не выполняющих модификацию полей), `noexcept` (для методов, не бросающих исключения).
6. В `main()` приложение должно работать в двух режимах:
   - [Обязательно] Тестовый пример: применить к задаче по поиску простых чисел «Решето Эратосфена» с использованием ваших реализованных Множеств и Битовых полей (2 способа: через функции битовых полей, через функции множеств).
   - [НЕ обязательно] Приложение Фильмотека с использованием ваших реализованных Множеств.

АЛЬТЕРНАТИВА СОЗДАНИЮ МОДУЛЕЙ. Если по какой-то причине вы не можете начать писать лабораторную, потому что не получается справиться с этим этапом – альтернатива: создать статические библиотеки, добавить зависимости, в настройках прописать пути – это мы учились делать, и вы это по умолчанию умеете.

Однако больше модулей не будет и это единственная работа в курсе, на которой вы сможете потренироваться в этом новшестве. Потому альтернатива крайне рекомендована.