**Міністерство освіти і науки України**
**Львівський національний університет імені Івана Франка**
**Факультет прикладної математики та інформатики**

Кафедра дискретного аналізу та інтелектуальних систем

Лабораторна робота №9

# БІТОВА МНОЖИНА

з курсу “Алгоритми та структури даних”

Виконав:
студент групи ПМІ-13
Лук’янчук Денис
Євгенійович

Львів – 2024

**Бітова множина** - це структура даних в програмуванні, що використовується для представлення множини елементів, де кожен елемент може бути присутнім або відсутнім. Кожен біт у бітовій множині відповідає певному елементу, і його значення вказує на те, чи належить цей елемент множині.

Бітові множини часто використовуються в програмуванні для ефективної реалізації різних операцій над множинами, таких як об'єднання, перетин, різниця тощо. Вони дозволяють економити пам'ять і швидше виконувати операції порівняно з іншими структурами даних, такими як масиви або списки.

**Алгоритм виконання**:

1. Створення класу BitSet:
   - Створив клас BitSet.
   - Описав приватне поле bits, яке буде представляти масив бітів.
   - Визначив константу MAX_SIZE для максимального розміру множини.

2. Реалізація методів для роботи з множинами:
   - Реалізував методи insert, remove та contains для додавання, видалення та перевірки наявності елементів у множині.
   - Використав методи set, reset та test з класу bitset для встановлення, скидання та перевірки бітів.

3. Реалізація операцій над множинами:
   - Додав методи unionWith, intersectWith та differenceWith для об'єднання, перетину та різниці множин відповідно.
   - Використав оператори `|`, `&` та `~` для виконання відповідних операцій над масивами бітів.

4. Реалізація тестового коду:
   - Створив тестові випадки, які перевіряють коректність роботи всіх методів класу BitSet, а також операцій над множинами.

5. Реалізація виводу результатів:
   - Додав метод display, який виводить масив бітів у зручному форматі.

6. Реалізація функції main:
   - Створив об'єкти класу `BitSet`.
   - Використав раніше реалізовані методи для виконання операцій над множинами.

**Приклад**

Дано: size = 10; set1 = {1, 2, 3}; set2 = {3, 4, 5}

```
Bitset: 0000001110
Bitset: 0000111000
Union of sets:
Bitset: 0000111110
Intersection of sets:
Bitset: 0000001000
Difference of sets:
Bitset: 0000000110
Does the set contain element 6? No
Does the set contain element 1? Yes
Bitset after remove:
Bitset: 0000000110
```

Приклад Unit-тесту(*без помилок*)

```
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from BitSetTest
[ RUN      ] BitSetTest.UnionTest
[       OK ] BitSetTest.UnionTest (0 ms)
[ RUN      ] BitSetTest.IntersectionTest
[       OK ] BitSetTest.IntersectionTest (0 ms)
[ RUN      ] BitSetTest.DifferenceTest
[       OK ] BitSetTest.DifferenceTest (0 ms)
[----------] 3 tests from BitSetTest (0 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (3 ms total)
[  PASSED  ] 3 tests.
```

Приклад Unit-тесту(*з помилкою*)

```
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from BitSetTest
[ RUN      ] BitSetTest.UnionTest
C:\└ыуюЁшЄьш Єр ёЄЁєъЄеЁш фрэшї\Test_9\Test_9\Test_9.cpp(74): error: Value
  Actual: false
Expected: true

[  FAILED  ] BitSetTest.UnionTest (1 ms)
[ RUN      ] BitSetTest.IntersectionTest
[       OK ] BitSetTest.IntersectionTest (0 ms)
[ RUN      ] BitSetTest.DifferenceTest
[       OK ] BitSetTest.DifferenceTest (0 ms)
[----------] 3 tests from BitSetTest (3 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (5 ms total)
[  PASSED  ] 2 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] BitSetTest.UnionTest

 1 FAILED TEST
```

**Висновок:** Під час написання класу BitSet для реалізації операцій над множинами було використано підхід, що базується на використанні масиву бітів. Кожен елемент множини відображений на відповідний біт у масиві, дозволяючи ефективно виконувати операції, такі як додавання, видалення та перевірка наявності елементів. Крім того, були реалізовані операції над множинами, такі як об'єднання, перетин та різниця, що дозволяють виконувати різноманітні операції з множинами. Цей підхід дозволяє створити простий та ефективний інструмент для роботи з множинами на основі бітів. З використанням класу BitSet можна легко вирішувати завдання, пов'язані з множинами, такі як визначення спільних елементів, виключення дублікатів та інші.