МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. Шухова»
(БГТУ им. В. Г. Шухова)

Кафедра программного обеспечения вычислительной
техники и автоматизированных систем

Лабораторная работа №1.1
по дисциплине: «Дискретная математика»
по теме: «**Операции над множествами**»

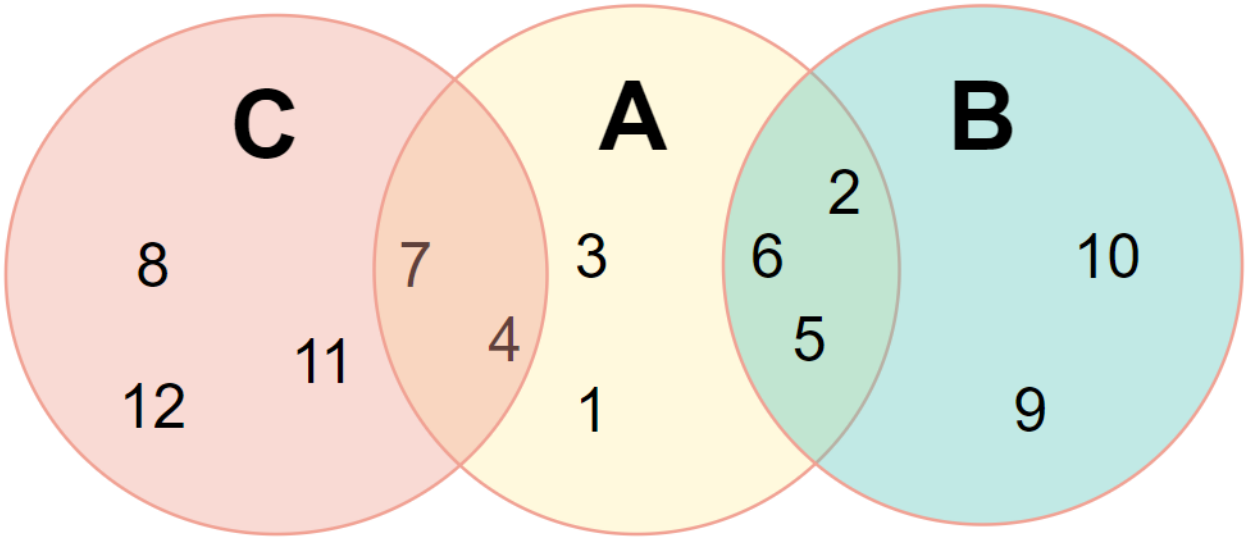
Выполнил/а: ст. группы ПВ-231
Чупахина София Александровна
Проверили:
Островский Алексей Мичеславович
Рязанов Юрий Дмитриевич

Белгород, 2024

Вариант 9

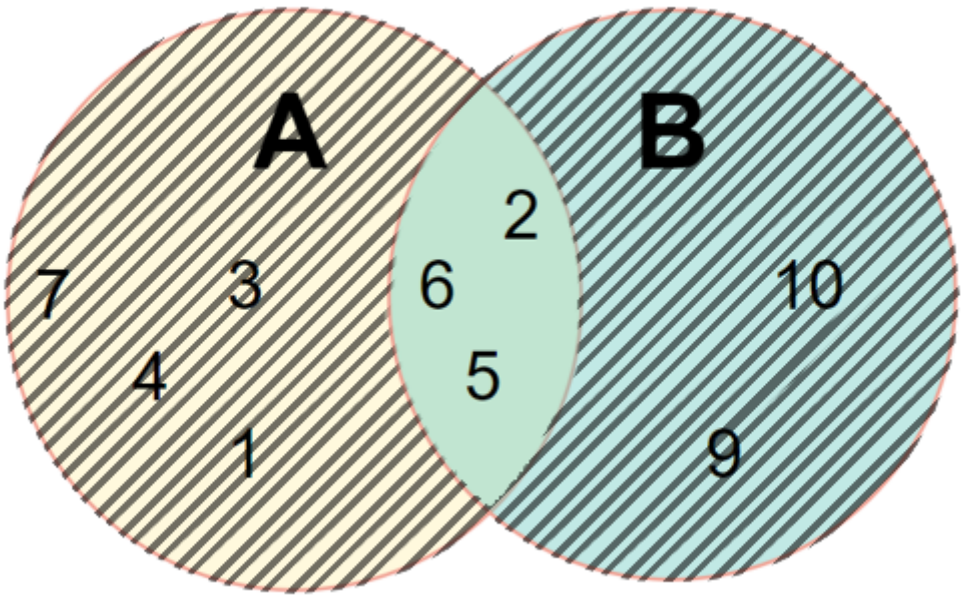
Задание 1. Вычислить значение выражения D = (A СИММР B) – C ИЛИ (C – A), где A = {1, 2, 3, 4, 5, 6, 7} B = {2, 5, 6, 9, 10} C = {4, 7, 8, 11, 12}. Считать U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Решение изобразить с помощью кругов Эйлера.

D = (A СИММР B) – C ИЛИ (C – A). На кругах Эйлера пересечение этих множеств выглядит так:

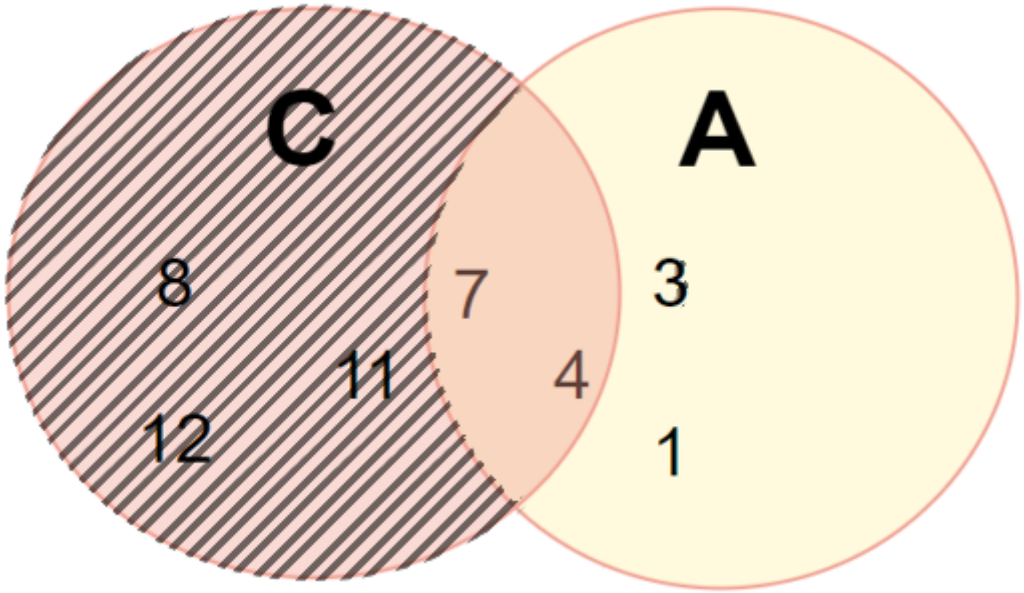


Все операции в этом выражении имеют одинаковый приоритет, поэтому сначала рассчитываем результат операций, заданных в скобках, слева направо, а потом остальные, также слева направо.

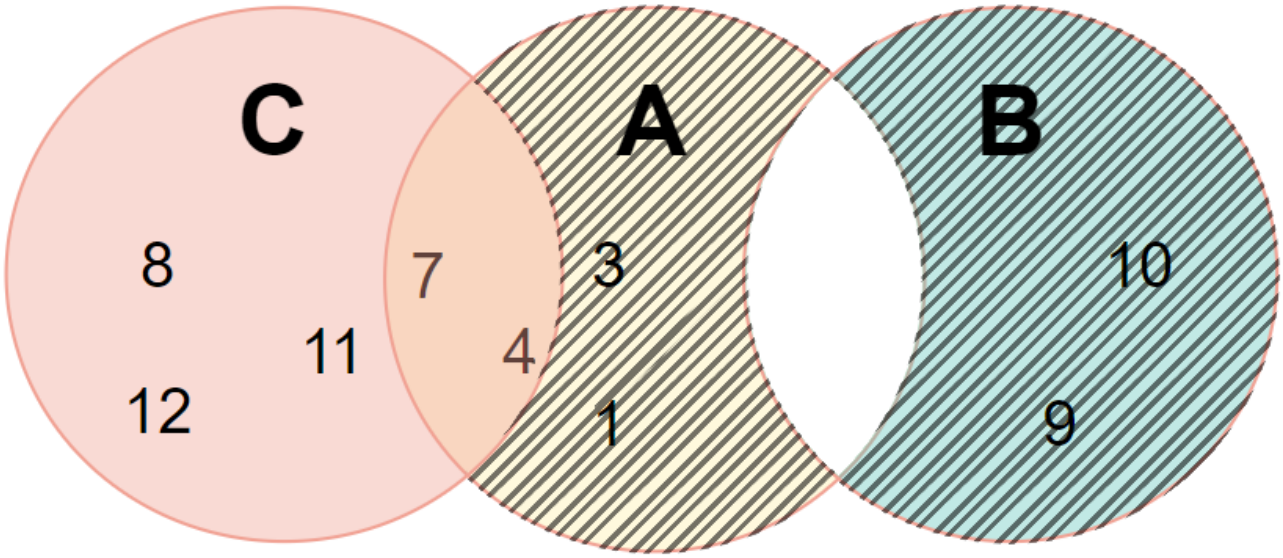
_1 = A СИММР B = {1, 2, 3, 4, 5, 6, 7} СИММР {2, 5, 6, 9, 10} = {1, 3, 4, 7, 9, 10}



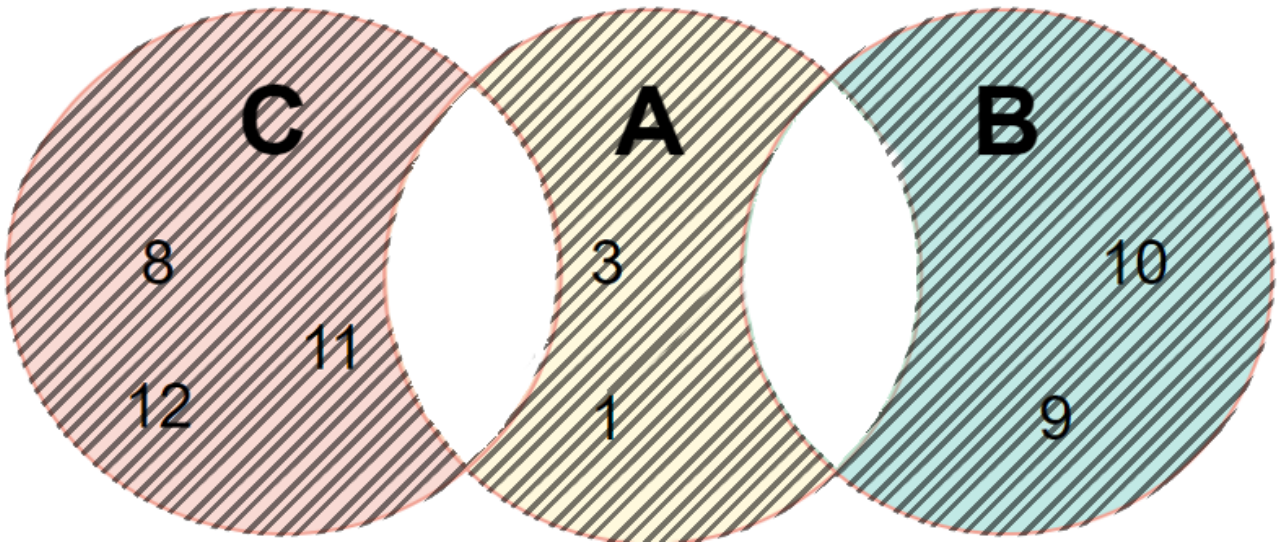
_2 = C – A = {4, 7, 8, 11, 12} – {1, 2, 3, 4, 5, 6, 7} = {8, 11, 12}



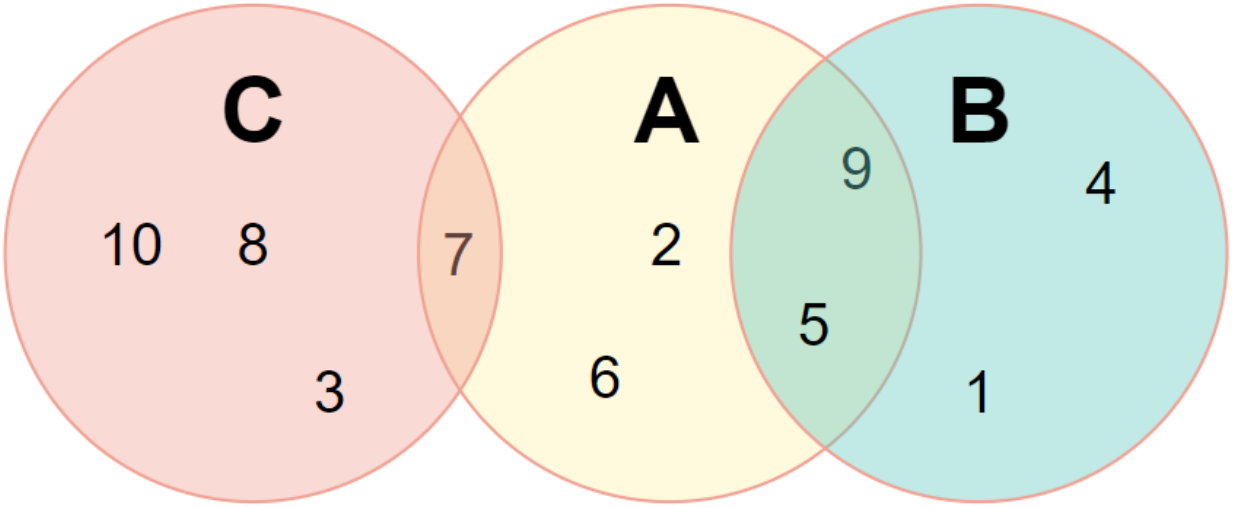
_3 = _1 – C = {1, 3, 4, 7, 9, 10} – {4, 7, 8, 11, 12} = {1, 3, 9, 10}



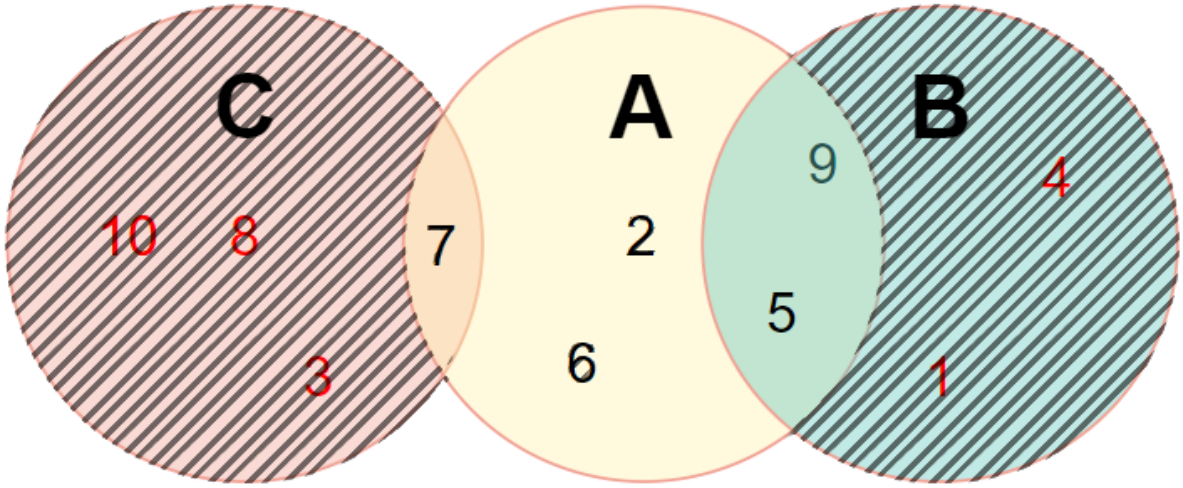
_4 = _3 ИЛИ _2 = {1, 3, 9, 10} ИЛИ {8, 11, 12} = {1, 3, 8, 9, 10, 11, 12}



Задание 2. Записать выражение в алгебре подмножеств, значение которого при заданных множествах A = {2, 5, 6, 7, 9} B = {1, 4, 5, 9} C = {3, 7, 8, 10} равно множеству D = {1, 3, 4, 8, 10}.
Изобразим множества A, B, C и их взаимное положение с помощью кругов Эйлера.



Далее выделим цветом те элементы, которые должны войти в итоговое множество D, и штриховкой – области, которые им принадлежат.



Легко заметить, что это те элементы множеств B и C, которые не принадлежат множеству A. Так что множество D можно получить в результате выражения (C ИЛИ B) – A

Задание 3. Программно реализовать операции над множествами, используя следующие способы представления множества в памяти ЭВМ:

а) элементы множества A хранятся в массиве A. Элементы массива A неупорядочены;
б) элементы множества A хранятся в массиве A. Элементы массива A упорядочены по возрастанию;
в) элементы множества A хранятся в массиве A, элементы которого типа boolean. Если i принадлежит A, то Ai=true, иначе Ai=false.

Подпункт а)

```c
#include <stdio.h>
#include <stdbool.h>
#define UNIVERSE_LENGTH 12

typedef struct {
    int set[UNIVERSE_LENGTH];
    size_t lenght;
} unordSet;

const unordSet UNIVERSUM = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12}, UNIVERSE_LENGTH};

unordSet join(unordSet set_a, unordSet set_b) {
    unordSet result = {{}, 0};
```

```
    for (int a_index = 0; a_index < set_a.lenght; a_index++) {
        result.set[a_index] = set_a.set[a_index];
    }
    result.lenght = set_a.lenght;

    for (int b_index = 0; b_index < set_b.lenght; b_index++) {
        bool is_element_unique = true;
        for (int a_index = 0; a_index < set_a.lenght; a_index++) {
            if (set_b.set[b_index] == set_a.set[a_index]) {
                is_element_unique = false;
                break;
            }
        }
        if (is_element_unique) {
            result.set[result.lenght] = set_b.set[b_index];
            result.lenght += 1;
        }
    }
    return result;
}

unordSet intersection(unordSet set_a, unordSet set_b) {
    unordSet result = {{}, 0};

    for (int a_index = 0; a_index < set_a.lenght; a_index++) {
        for (int b_index = 0; b_index < set_b.lenght; b_index++) {
            if (set_a.set[b_index] == set_b.set[a_index]) {
                result.set[result.lenght] = set_a.set[b_index];
                break;
            }
        }
    }
    return result;
}

unordSet difference(unordSet set_a, unordSet set_b) {
    unordSet result = {{}, 0};

    for (int a_index = 0; a_index < set_a.lenght; a_index++) {
        bool is_element_unique = true;
        for (int b_index = 0; b_index < set_b.lenght; b_index++) {
            if (set_a.set[b_index] == set_b.set[a_index]) {
                is_element_unique = false;
                break;
            }
        }
```

```
        if (is_element_unique) {
            result.set[result.lenght] = set_b.set[a_index];
            result.lenght += 1;
        }
    }
    return result;
}

unordSet symmetricalDifference(unordSet set_a, unordSet set_b) {
    unordSet a_minus_b = difference(set_a, set_b);
    unordSet b_minus_a = difference(set_b, set_a);
    return join(a_minus_b, b_minus_a);
}

unordSet addition(unordSet set_a) {
    return difference(UNIVERSUM, set_a);
}

bool inclusion(unordSet set_a, unordSet set_b) {
    bool is_b_part_of_a = true;
    for (int b_index = 0; b_index < set_b.lenght; b_index++) {
        bool is_element_unique = true;
        for (int a_index = 0; a_index < set_a.lenght; a_index++) {
            if (set_a.set[b_index] == set_b.set[a_index]) {
                is_element_unique = false;
                break;
            }
        }
        if (is_element_unique) {
            is_b_part_of_a = false;
            break;
        }
    }
    return is_b_part_of_a;
}

bool strictInclusion(unordSet set_a, unordSet set_b) {
    return inclusion(set_a, set_b) && (set_a.lenght !=
set_b.lenght);
}

bool equality(unordSet set_a, unordSet set_b) {
    return inclusion(set_a, set_b) && (set_a.lenght ==
set_b.lenght);
}
```

Подпункт в)

```c
#include <stdio.h>
#include <stdbool.h>
#define UNIVERSE_LENGTH 12

typedef struct {
    bool set[UNIVERSE_LENGTH];
} bitSet;

const bitSet UNIVERSUM = {{true, true, true, true, true, true,
true, true, true, true, true, true}};

bitSet join(bitSet set_a, bitSet set_b) {
    bitSet result = {{}};
    for (int index = 0; index < UNIVERSE_LENGTH; index++) {
        result.set[index] = set_a.set[index] || set_b.set[index];
    }
    return result;
}

bitSet intersection(bitSet set_a, bitSet set_b) {
    bitSet result = {{}};
    for (int index = 0; index < UNIVERSE_LENGTH; index++) {
        result.set[index] = set_a.set[index] && set_b.set[index];
    }
    return result;
}

bitSet difference(bitSet set_a, bitSet set_b) {
    bitSet result = {{}};
    for (int index = 0; index < UNIVERSE_LENGTH; index++) {
        result.set[index] = set_a.set[index] && !set_b.set[index];
    }
    return result;
}

bitSet symmetricalDifference(bitSet set_a, bitSet set_b) {
    bitSet result = {{}};
    for (int index = 0; index < UNIVERSE_LENGTH; index++) {
        result.set[index] = set_a.set[index] ^ set_b.set[index];
    }
    return result;
}

bitSet addition(bitSet set_a) {
```

```
    return difference(UNIVERSUM, set_a);
}

bool inclusion(bitSet set_a, bitSet set_b) {
    bool is_b_part_of_a = true;
    for (int index = 0; index < UNIVERSE_LENGTH; index++) {
        if (!set_a.set[index] && set_b.set[index]) {
            is_b_part_of_a = false;
            break;
        }
    }
    return is_b_part_of_a;
}

bool equality(bitSet set_a, bitSet set_b) {
    bool are_sets_equal = 1;
    for (int index = 0; index < UNIVERSE_LENGTH; index++) {
        if (set_a.set[index] != set_b.set[index]) {
            are_sets_equal = false;
            break;
        }
    }
    return are_sets_equal;
}

bool strictInclusion(bitSet set_a, bitSet set_b) {
    return inclusion(set_a, set_b) && !equality(set_a, set_b);
}
```