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)

Институт энергетики, информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 1.1
по дисциплине дискретная математика
тема: Операции над множествами

Выполнил: студент группы ПВ-223
Игнатьев Артур Олегович
Проверил: доцент
Рязанов Юрий Дмитриевич
старший преподаватель
Бондаренко Татьяна Владимировна

Белгород 2022

Лабораторная работа № 1.1

Тема: Операции над множествами

Цель работы: изучить и научиться использовать алгебру подмножеств, изучить различные способы представления множеств в памяти ЭВМ, научиться программно реализовывать операции над множествами и выражения в алгебре подмножеств.

Задания

1. Вычислить значение выражения (см. “Варианты заданий”, п. а). Во всех вариантах считать U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Решение изобразить с помощью кругов Эйлера.

2. Записать выражение в алгебре подмножеств, значение которого при заданных множествах A, B и C равно множеству D (см. “Варианты заданий”, п. б).

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 $i \in A$, то A_i=true, иначе A_i=false.

4. Написать программы для вычисления значений выражений (см. “Задания”, п.1 и п.2).

5. Используя программы (см. “Задания”, п.4), вычислить значения

выражений (см. “Задания”, п.1 и п.2).

Вариант 3

а) D=(A – (BΔC)) ⋃ ((BΔC) – A)

A={1,2,4,5,8} B={2,3,5,6,9} C={4,5,6,7,9}

б) A={2,3,4,5,6} B={1,2,4,9} C={4,5,7,8}

D={3,6}

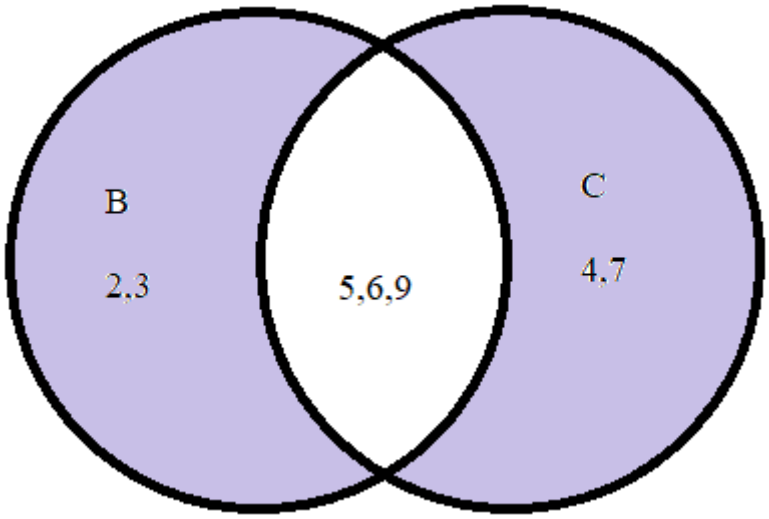
Решение

1. Вычислить значение выражения (см. “Варианты заданий”, п. а). Во всех вариантах считать U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Решение изобразить с помощью кругов Эйлера.

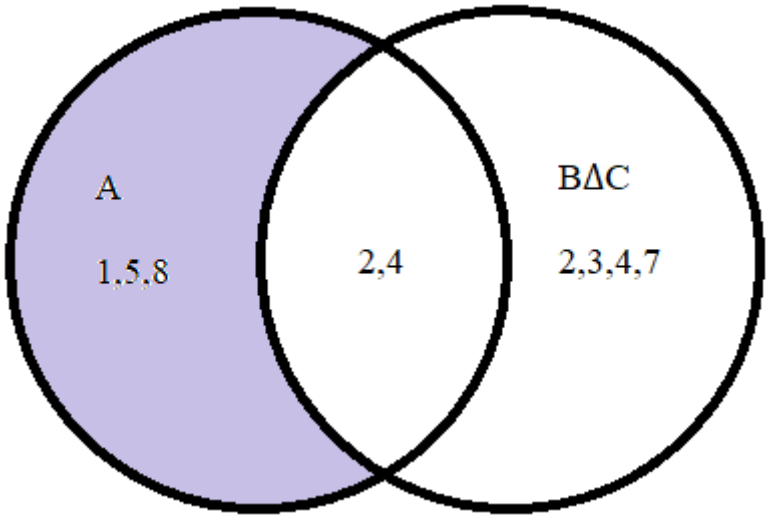
а) D = (A – (BΔC)) ⋃ ((BΔC) – A)

A = {1,2,4,5,8} B = {2,3,5,6,9} C = {4,5,6,7,9}

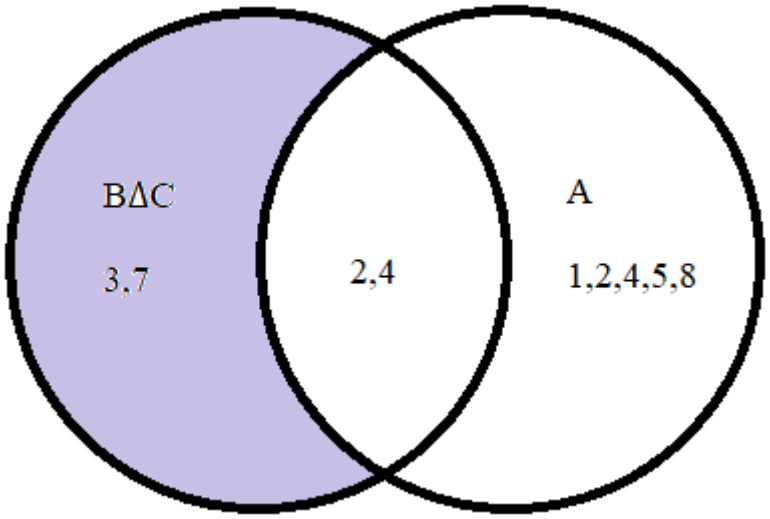
1)= BΔC = {2,3,5,6,9} Δ {4,5,6,7,9} = {2,3,4,7}



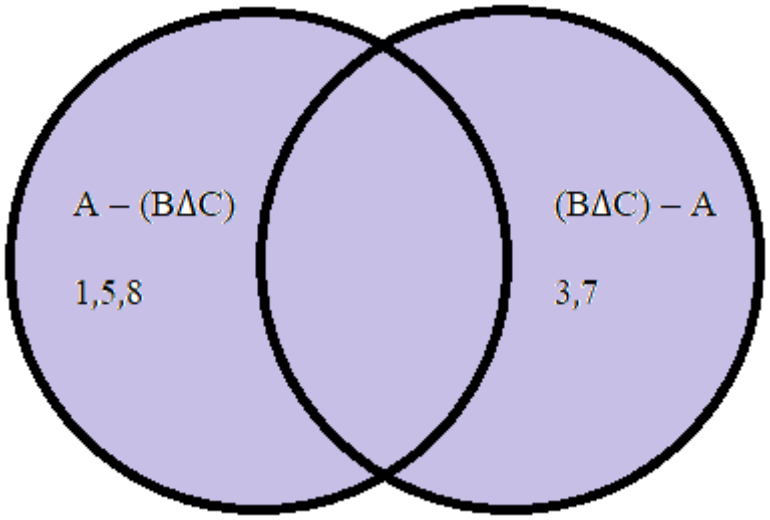
2)=A - (BΔC) = {1,2,4,5,8} - {2,3,4,7} = {1,5,8}

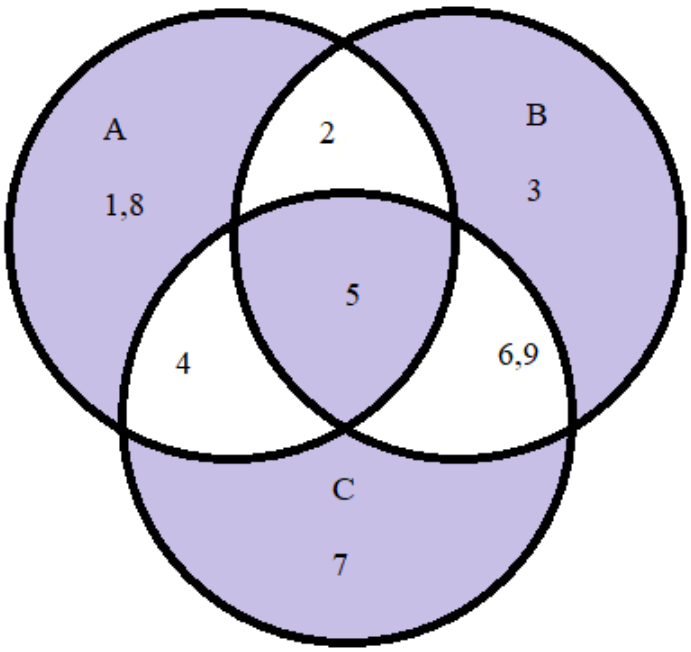


3) = (BΔC) – A) = {2,3,4,7} - {1,2,4,5,8} = {3,7}



4) D = (A – (BΔC)) ∪ ((BΔC) – A) = {1,5,8} ∪ {3,7} = {1,3,5,7,8}




A
1,8
2
B
3
5
4
6,9
C
7

2. Записать выражение в алгебре подмножеств, значение которого при заданных множествах A, B и C равно множеству D (см. “Варианты заданий”, п. б).

б) A={2,3,4,5,6} B={1,2,4,9} C={4,5,7,8}

D={3,6}

Первый способ

1) A-B={2,3,4,5,6}-{1,2,4,9}={3,5,6}
2) (A-B)-C={3,5,6}-{4,5,7,8}={3,6}

D=(A-B)-C={3,6}

Второй способ

1) A-C={2,3,4,5,6}-{4,5,7,8}={2,3,6}
2) (A-C)-B={2,3,6}-{1,2,4,9}={3,6}

D=(A-C)-B={3,6}

При решении данного задания, я обратил внимание на некоторые элементы множества A∈D и на их основе вывел два способа получения множества D.

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∈A, то Ai=true, иначе Ai=false.

а) Операция объединение

```
#include <stdbool.h>

// Осуществляет операцию объединения над массивами A и B
записывает результат в массив C
void Union(const int *const arrayA,
           const int *const arrayB,
           int *const arrayC) {
    for (int i = 1; i <= arrayA[0]; i++) {
        arrayC[i] = arrayA[i];
    }
    arrayC[0] = arrayA[0];

    int elementsArrayC = arrayC[0];
    bool elementsEqual = false;
    for (int i = 1; i <= arrayB[0]; i++) {
        for (int n = 1; n <= elementsArrayC; n++) {
            if (arrayB[i] == arrayC[n]) {
                elementsEqual = true;
            }
        }
        if (!elementsEqual) {
            arrayC[arrayC[0] + 1] = arrayB[i];
            arrayC[0]++;
        } else {
            elementsEqual = false;
        }
    }
}
```

Операция пересечение

```
#include <stdbool.h>

// Осуществляет операцию пересечения над массивами A и B
записывает результат в массив C
void intersection(const int *const arrayA,
                  const int *const arrayB,
                  int *const arrayC) {
    arrayC[0] = 0;
    bool elementsEqual = false;
    for (int i = 1; i <= arrayA[0]; i++) {
        for (int n = 1; n <= arrayB[0]; n++) {
            if (arrayA[i] == arrayB[n]) {
                elementsEqual = true;
            }
        }
        if (elementsEqual) {
            arrayC[arrayC[0] + 1] = arrayA[i];
            arrayC[0]++;
            elementsEqual = false;
        }
    }
}
```

Операция разность

```
#include <stdbool.h>

// Осуществляет операцию разности над массивами A и B записывает
результат в массив C
void difference(const int *const arrayA,
                const int *const arrayB,
                int *const arrayC) {
    arrayC[0] = 0;
    bool elementsEqual = false;
    for (int i = 1; i <= arrayA[0]; i++) {
        for (int n = 1; n <= arrayB[0]; n++) {
            if (arrayA[i] == arrayB[n]) {
                elementsEqual = true;
            }
        }
        if (!elementsEqual) {
            arrayC[arrayC[0] + 1] = arrayA[i];
            arrayC[0]++;
        } else {
            elementsEqual = false;
        }
    }
}
```

Операция симметрическая разность

```c
#include <stdbool.h>

// Проверяет есть ли число num в массиве array
bool numArray(int num, const int *const array) {
    for (int i = 1; i <= array[0]; i++) {
        if (num == array[i]) {
            return true;
        }
    }
    return false;
}

// Выполняет операцию симметрической разности над массивами A и
B и результат записывает в массив C
void symmetricalDifference(const int *const arrayA,
                           const int *const arrayB,
                           int *const arrayC) {
    arrayC[0] = 0;
    for (int i = 1; i <= arrayA[0]; i++) {
        if (!numArray(arrayA[i], arrayB)) {
            arrayC[arrayC[0] + 1] = arrayA[i];
            arrayC[0]++;
        }
    }
    for (int i = 1; i <= arrayB[0]; i++) {
        if (!numArray(arrayB[i], arrayA)) {
            arrayC[arrayC[0] + 1] = arrayB[i];
            arrayC[0]++;
        }
    }
}
```

Операция дополнение:

```
// Максимальный размер массива A
#define MAX_SIZE 10

// Функция для проверки, содержит ли массив элемент key
int contains(int arr[], int size, int key) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            return 1; // Возвращаем 1, если элемент найден
        }
    }
    return 0; // Возвращаем 0, если элемент не найден
}

// Функция для вычисления дополнения множества A в массиве A
void complement(int arrA[], int sizeA, int arrComplement[], int
*sizeComplement) {
    // Предполагаем, что массив arrComplement уже имеет
достаточный размер
    *sizeComplement = 0; // Инициализируем размер дополнения
множества A

    // Проверяем каждое возможное значение от 1 до MAX_SIZE
    for (int i = 1; i <= MAX_SIZE; i++) {
        // Если текущее значение не содержится в массиве arrA,
добавляем его в массив дополнения
        if (!contains(arrA, sizeA, i)) {
            arrComplement[*sizeComplement] = i;
            (*sizeComplement)++;
        }
    }
}
```

б) Операция объединение

```
// Выполняет операцию объединения над массивами A и B и
записывает результат в массив C
void unite(const int *const arrayA,
           const int *const arrayB,
           int *const arrayC) {
    int breakA = 0;
    int breakB = 0;
    int actualElementC = 1;
    int breakPoint = 0;
    int statusI;
    for (int i = 1; i <= arrayA[0] + arrayB[0]; i++) {
        if (arrayA[0] < i - breakA || arrayB[0] < i - breakB) {
            breakPoint = (arrayA[0] + breakA < i ? 'A' : 'B');
            statusI = i;
            break;
        }
        if (arrayA[i - breakA] == arrayB[i - breakB]) {
            arrayC[actualElementC] = arrayA[i - breakA];
            actualElementC++;
        } else if (arrayA[i - breakA] > arrayB[i - breakB]) {
            arrayC[actualElementC] = arrayB[i - breakB];
            breakA++;
            actualElementC++;
        } else if (arrayA[i - breakA] < arrayB[i - breakB]) {
            arrayC[actualElementC] = arrayA[i - breakA];
            breakB++;
            actualElementC++;
        }
    }
    if (breakPoint == 'B') {
        for (int i = statusI - breakA; i <= arrayA[0]; i++) {
            arrayC[actualElementC] = arrayA[i];
            actualElementC++;
        }
    } else if (breakPoint == 'A') {
        for (int i = statusI - breakB; i <= arrayB[0]; i++) {
            arrayC[actualElementC] = arrayB[i];
            actualElementC++;
        }
    }
    arrayC[0] = actualElementC -= 1;
}
```

Операция пересечение

```
// Выполняет операцию пересечения над массивами A и B и
записывает результат в массив C
void intersection(const int *const arrayA,
                  const int *const arrayB,
                  int *const arrayC) {
    arrayC[0] = 0;
    int elementNumber = 1;
    for (int i = 1; i <= arrayB[0]; i++) {
        if (elementNumber > arrayA[0] || i > arrayB[0]) {
            break;
        }
        if (arrayA[elementNumber] == arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayB[i];
            elementNumber++;
        } else if (arrayA[elementNumber] < arrayB[i]) {
            elementNumber++;
            i--;
        }
    }
}
```

Операция разность

```
// Выполняет операцию разности над массивами A и B и записывает
результат в массив C
void difference(const int *const arrayA,
                const int *const arrayB,
                int *const arrayC) {
    arrayC[0] = 0;
    int elementNumber = 1;
    for (int i = 1; elementNumber <= arrayA[0]; i++) {
        if (arrayA[elementNumber] == arrayB[i]) {
            elementNumber++;
        } else if (arrayA[elementNumber] < arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementNumber];
            elementNumber++;
            i--;
        } else if (i == arrayB[0] && arrayA[elementNumber] >
arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementNumber];
            elementNumber++;
        }
        if (i + 1 > arrayB[0]) {
            i--;
        }
    }
}
```

Операция симметрическая разность

```
#include <stdbool.h>

// Выполняет операцию симметрической разности над массивами A и
B записывает результат в массив C
void symmetricalDifference(const int *const arrayA,
                           const int *const arrayB,
                           int *const arrayC) {
    arrayC[0] = 0;
    int elementArrayA = 1;
    int elementArrayB = 1;
    while (arrayA[0] > elementArrayA && arrayB[0] >
elementArrayB) {
        if (arrayA[0] < elementArrayA) {
            elementArrayA--;
        } else if (arrayB[0] < elementArrayB) {
            elementArrayB--;
        }
        if (arrayA[elementArrayA] < arrayB[elementArrayB]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementArrayA];
            elementArrayA++;
        } else if (arrayA[elementArrayA] >
arrayB[elementArrayB]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayB[elementArrayB];
            elementArrayB++;
        } else {
            elementArrayA++;
            elementArrayB++;
        }
    }
    bool impact = false;
    bool numberDetected = false;
    if (arrayA[0] - elementArrayA < arrayB[0] - elementArrayB) {
        for (int i = elementArrayB; i <= arrayB[0]; i++) {
            if (arrayB[i] == arrayA[elementArrayA]) {
                numberDetected = true;
            } else if (arrayB[i] > arrayA[elementArrayA] &&
                       !numberDetected && !impact) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayA[elementArrayA];
                impact = true;
            }
            if (arrayB[i] != arrayA[elementArrayA]) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayB[i];
            }
        }
    } else {
        for (int i = elementArrayB; i <= arrayB[0]; i++) {
            if (arrayA[i] == arrayB[elementArrayB]) {
```

```
                numberDetected = true;
            } else if (arrayA[i] > arrayB[elementArrayB] &&
                       !numberDetected && !impact) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayB[elementArrayB];
                impact = true;
            }
            if (arrayA[i] != arrayB[elementArrayB]) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayB[i];
            }
        }
    }
}
```

Операция дополнение:

```
void complement(int A[], int sizeA, int universe[], int
sizeUniverse) {
    int complement[sizeUniverse];
    int i, j, k;
    j = 0;  // Индекс для прохода по массиву множества A
    k = 0;  // Индекс для прохода по массиву дополнения

    // Обходим универсум и проверяем, присутствует ли каждый
элемент в множестве A
    for (i = 0; i < sizeUniverse; i++) {
        if (j < sizeA && universe[i] == A[j]) {
            j++;  // Если элемент присутствует в множестве A,
переходим к следующему элементу
        } else {
            complement[k] = universe[i];  // Если элемент
отсутствует, добавляем его в дополнение
            k++;
        }
    }
}
```

в) Операция объединение

```
// Выполняет операцию объединения над массивами A и B и
записывает результат в массив C
void unite(const int *const arrayA,
           const int *const arrayB,
           int *const arrayC, int arraySize) {
    for (int i = 0; i < arraySize; i++) {
        arrayC[i] = arrayA[i] || arrayB[i];
    }
}
```

Операция пересечение

```
// Выполняет операцию пересечения над массивами A и B и
записывает результат в массив C
void intersection(const int *const arrayA,
                  const int *const arrayB,
                  int *const arrayC, int arraysSize) {
    for (int i = 0; i < arraysSize; i++) {
        arrayC[i] = arrayA[i] && arrayB[i];
    }
}
```

Операция разность

```
// Выполняет операцию разности над массивами A и B и записывает
результат в массив C
void difference(const int *const arrayA,
                const int *const arrayB,
                int *const arrayC, int arraysSize) {
    for (int i = 0; i < arraysSize; i++) {
        arrayC[i] = arrayA[i] && !arrayB[i];
    }
}
```

Операция симметричная разность

```
// Выплняет операцию симметрической разности над массивами A и B
и записывает результат в массив C
void symmetricalDifference(const int *const arrayA, const int
*const arrayB,
                           int *const arrayC, int arraysSize) {
    for (int i = 0; i < arraysSize; i++) {
        arrayC[i] = arrayA[i] && !arrayB[i] || !arrayA[i] &&
arrayB[i];
    }
}
```

Операция дополнение:

```
// Функция для вычисления дополнения множества
void complement(bool A[], int size) {
    for (int i = 0; i < size; i++) {
        A[i] = !A[i];  // Инвертируем значение элемента
    }
}
```

4. Написать программы для вычисления значений выражений (см. “Задания”, п.1 и п.2).

Программа для вычисления из задания 1

```c
#include <stdio.h>
#include <stdbool.h>
#include <windows.h>

// Выполняет операцию объединения над массивами A и B и
записывает результат в массив C
void unite(const int *const arrayA,
           const int *const arrayB,
           int *const arrayC) {
    int breakA = 0;
    int breakB = 0;
    int actualElementC = 1;
    int breakPoint = 0;
    int statusI;
    for (int i = 1; i <= arrayA[0] + arrayB[0]; i++) {
        if (arrayA[0] < i - breakA || arrayB[0] < i - breakB) {
            breakPoint = (arrayA[0] + breakA < i ? 'A' : 'B');
            statusI = i;
            break;
        }
        if (arrayA[i - breakA] == arrayB[i - breakB]) {
            arrayC[actualElementC] = arrayA[i - breakA];
            actualElementC++;
        } else if (arrayA[i - breakA] > arrayB[i - breakB]) {
            arrayC[actualElementC] = arrayB[i - breakB];
            breakA++;
            actualElementC++;
        } else if (arrayA[i - breakA] < arrayB[i - breakB]) {
            arrayC[actualElementC] = arrayA[i - breakA];
            breakB++;
            actualElementC++;
        }
    }
    if (breakPoint == 'B') {
        for (int i = statusI - breakA; i <= arrayA[0]; i++) {
            arrayC[actualElementC] = arrayA[i];
            actualElementC++;
        }
    } else if (breakPoint == 'A') {
        for (int i = statusI - breakB; i <= arrayB[0]; i++) {
            arrayC[actualElementC] = arrayB[i];
            actualElementC++;
        }
    }
    arrayC[0] = actualElementC -= 1;
}
```

```
// Выполняет операцию пересечения над массивами A и B и
записывает результат в массив C
void intersection(const int *const arrayA,
                  const int *const arrayB,
                  int *const arrayC) {
    arrayC[0] = 0;
    int elementNumber = 1;
    for (int i = 1; i <= arrayB[0]; i++) {
        if (elementNumber > arrayA[0] || i > arrayB[0]) {
            break;
        }
        if (arrayA[elementNumber] == arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayB[i];
            elementNumber++;
        } else if (arrayA[elementNumber] < arrayB[i]) {
            elementNumber++;
            i--;
        }
    }
}

// Выполняет операцию разности над массивами A и B и записывает
результат в массив C
void difference(const int *const arrayA,
                const int *const arrayB,
                int *const arrayC) {
    arrayC[0] = 0;
    int elementNumber = 1;
    for (int i = 1; elementNumber <= arrayA[0]; i++) {
        if (arrayA[elementNumber] == arrayB[i]) {
            elementNumber++;
        } else if (arrayA[elementNumber] < arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementNumber];
            elementNumber++;
            i--;
        } else if (i == arrayB[0] && arrayA[elementNumber] >
arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementNumber];
            elementNumber++;
        }
        if (i + 1 > arrayB[0]) {
            i--;
        }
    }
}

// Выполняет операцию симметрической разности над массивами A и
B записывает результат в массив C
void symmetricalDifference(const int *const arrayA,
                           const int *const arrayB,
```

```
                            int *const arrayC) {
    arrayC[0] = 0;
    int elementArrayA = 1;
    int elementArrayB = 1;
    while (arrayA[0] > elementArrayA && arrayB[0] >
elementArrayB) {
        if (arrayA[0] < elementArrayA) {
            elementArrayA--;
        } else if (arrayB[0] < elementArrayB) {
            elementArrayB--;
        }
        if (arrayA[elementArrayA] < arrayB[elementArrayB]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementArrayA];
            elementArrayA++;
        } else if (arrayA[elementArrayA] >
arrayB[elementArrayB]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayB[elementArrayB];
            elementArrayB++;
        } else {
            elementArrayA++;
            elementArrayB++;
        }
    }
    bool impact = false;
    bool numberDetected = false;
    if (arrayA[0] - elementArrayA < arrayB[0] - elementArrayB) {
        for (int i = elementArrayB; i <= arrayB[0]; i++) {
            if (arrayB[i] == arrayA[elementArrayA]) {
                numberDetected = true;
            } else if (arrayB[i] > arrayA[elementArrayA] &&
                       !numberDetected && !impact) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayA[elementArrayA];
                impact = true;
            }
            if (arrayB[i] != arrayA[elementArrayA]) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayB[i];
            }
        }
    } else {
        for (int i = elementArrayB; i <= arrayB[0]; i++) {
            if (arrayA[i] == arrayB[elementArrayB]) {
                numberDetected = true;
            } else if (arrayA[i] > arrayB[elementArrayB] &&
                       !numberDetected && !impact) {
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayB[elementArrayB];
                impact = true;
            }
            if (arrayA[i] != arrayB[elementArrayB]) {
```

```
                arrayC[0]++;
                arrayC[arrayC[0]] = arrayB[i];
            }
        }
    }
}

int main() {
    SetConsoleOutputCP(CP_UTF8);

    int arrayA[10];
    int arrayB[10];
    int arrayC[10];
    int res1[30];
    int res2[30];
    int res3[30];

// Ввод массива A
    printf("Введите размер массива A \n");
    scanf("%d", &arrayA[0]);
    printf("Введите элементы массива A \n");
    for (int i = 1; i <= arrayA[0]; i++) {
        scanf("%d", &arrayA[i]);
    }

// Ввод массива B
    printf("Введите размер массива B \n");
    scanf("%d", &arrayB[0]);
    printf("Введите элементы массива B \n");
    for (int i = 1; i <= arrayB[0]; i++) {
        scanf("%d", &arrayB[i]);
    }

// Ввод массива C
    printf("Введите размер массива C \n");
    scanf("%d", &arrayC[0]);
    printf("Введите элементы массива C \n");
    for (int i = 1; i <= arrayC[0]; i++) {
        scanf("%d", &arrayC[i]);
    }

// Подсчёт значения выражения
    symmetricalDifference(arrayB, arrayC, res1);
    difference(arrayA, res1, res2);
    difference(res1, arrayA, res3);
    res1[0] = 0;
    unite(res2, res3, res1);

// Вывод результатов
    printf("Результат \n");
    for (int i = 1; i <= res1[0]; i++) {
        printf("%d ", res1[i]);
    }
```

```
    return 0;
}
```

Программа для вычисления из задания 2

```
#include <stdio.h>
#include <windows.h>

// Выполняет операцию разности над массивами A и B и записывает
результат в массив C
void difference(const int *const arrayA,
                const int *const arrayB,
                int *const arrayC) {
    arrayC[0] = 0;
    int elementNumber = 1;
    for (int i = 1; elementNumber <= arrayA[0]; i++) {
        if (arrayA[elementNumber] == arrayB[i]) {
            elementNumber++;
        } else if (arrayA[elementNumber] < arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementNumber];
            elementNumber++;
            i--;
        } else if (i == arrayB[0] && arrayA[elementNumber] >
arrayB[i]) {
            arrayC[0]++;
            arrayC[arrayC[0]] = arrayA[elementNumber];
            elementNumber++;
        }
        if (i + 1 > arrayB[0]) {
            i--;
        }
    }
}

int main() {
    SetConsoleOutputCP(CP_UTF8);

    int arrayA[10];
    int arrayB[10];
    int arrayC[10];
    int res1[30];
    int res2[30];

// Ввод массива A
    printf("Введите размер массива A \n");
    scanf("%d", &arrayA[0]);
    printf("Введите элементы массива A \n");
    for (int i = 1; i <= arrayA[0]; i++) {
        scanf("%d", &arrayA[i]);
    }
```

```
// Ввод массива B
    printf("Введите размер массива B \n");
    scanf("%d", &arrayB[0]);
    printf("Введите элементы массива B \n");
    for (int i = 1; i <= arrayB[0]; i++) {
        scanf("%d", &arrayB[i]);
    }

// Ввод массива C
    printf("Введите размер массива C \n");
    scanf("%d", &arrayC[0]);
    printf("Введите элементы массива C \n");
    for (int i = 1; i <= arrayC[0]; i++) {
        scanf("%d", &arrayC[i]);
    }

// Подсчёт значения выражения
    difference(arrayA, arrayB, res1);
    difference(res1, arrayC, res2);

// Вывод результатов
    printf("Результат \n");
    for (int i = 1; i <= res2[0]; i++) {
        printf("%d ", res2[i]);
    }
    return 0;
}
```

5. Используя программы (см. “Задания”, п.4), вычислить значения выражений (см. “Задания”, п.1 и п.2).

Результат из задания 1

```
Введите размер массива A
5
Введите элементы массива A
1 2 4 5 8
Введите размер массива B
5
Введите элементы массива B
2 3 5 6 9
Введите размер массива C
5
Введите элементы массива C
4 5 6 7 9
Результат
1 3 5 7 8
```

Результат из задания 2

```
Введите размер массива A
5
Введите элементы массива A
2 3 4 5 6
Введите размер массива B
4
Введите элементы массива B
1 2 4 9
Введите размер массива C
4
Введите элементы массива C
4 5 7 8
Результат
3 6
```

Вывод: на этой лабораторной работе я изучил и научился использовать алгебру подмножеств, изучил различные способы представления множеств в памяти ЭВМ, научился программно реализовывать операции над множествами и выражения в алгебре подмножеств.