

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования «Санкт-Петербургский
политехнический университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01. Математика и компьютерные науки

Дискретная математика

Отчет о выполнении лабораторной работы №1 Реализация генерации бинарного кода Грея и операций над мультимножествами на его основе

Студент Козловская О. А.
группы 5130201/30002

Преподаватель Востров А. В.

Санкт-Петербург, 2025

Содержание

Введение	4
1 Математическое описание	5
1.1 Мультимножества	5
1.2 Операции над множествами	5
1.2.1 Объединение	5
1.2.2 Пересечение	5
1.2.3 Разность	6
1.2.4 Симметрическая разность	6
1.2.5 Дополнение	6
1.3 Операции над мультимножествами	6
1.3.1 Объединение	6
1.3.2 Пересечение	6
1.3.3 Разность	7
1.3.4 Симметрическая разность	7
1.3.5 Дополнение	7
1.3.6 Арифметическая сумма	7
1.3.7 Арифметическая разность	7
1.3.8 Арифметическое произведение	7
1.3.9 Арифметическое деление	7
1.4 Построение бинарного кода Грея	8
2 Особенности реализации	9
2.1 Структуры данных класса	9
2.2 Функции класса	9
2.2.1 Генерация кода Грея	9
2.2.2 Установка количества элементов универсума	10
2.2.3 Заполнение мультимножеств	10
2.2.4 Объединение мультимножеств	11
2.2.5 Пересечение мультимножеств	11
2.2.6 Разность $A \setminus B$	11
2.2.7 Разность $B \setminus A$	12
2.2.8 Симметрическая разность	13
2.2.9 Дополнение A	13
2.2.10 Дополнение B	14
2.2.11 Арифметическая сумма	15
2.2.12 Арифметическая разность A - B	15
2.2.13 Арифметическая разность B - A	16

2.2.14	Арифметическое произведение	16
2.2.15	Функция арифметического деления	17
3	Результаты работы	18
3.1	Ввод универсума	18
3.2	Ввод множеств	18
3.3	Вывод результатов	19
3.4	Некорректный ввод	22
3.5	Ввод пустого универсума	22
	Заключение	23
	Список использованной литературы	25

Введение

В данной лабораторной работе требуется реализовать программу генерации бинарного кода Грея для заполнения универсума мультимножеств (заданной пользователем разрядности). На основе универсума формируются два мультимножества двумя способами заполнения - вручную и автоматически (выбирает пользователь). Мощности множеств задает пользователь. В результате на экран выводятся результаты действий над множествами: объединение, пересечение, разность (оба варианта), симметрическая разность, дополнение (оба варианта), арифметические сумма, разность, произведение и деление. Кроме того, требуется реализовать защиту от некорректного пользовательского ввода и пользовательское меню.

1 Математическое описание

Множества

Множество - это любая определённая совокупность объектов.

Элементы множества - объекты, из которых составлено множество.

Элементы множества различны и отличимы друг от друга. Как множествам, так и элементам можно давать имена или присваивать символьные обозначения.

Множество, не содержащее элементов, называется пустым(\emptyset).

1.1 Мультимножества

Мультимножества - совокупности элементов, в которые элементы входят по несколько раз. Пусть $X = \{x_1, \dots, x_n\}$ — некоторое (конечное) множество и пусть a_1, \dots, a_n — неотрицательные целые числа. Тогда мультимножеством \hat{X} (над множеством X) называется совокупность элементов множества X , в которую элемент x_i входит a_i раз, $a_i \geq 0$.

Мультимножество обозначается одним из следующих способов:
$$\hat{X} = [x_1^{a_1}, \dots, x_n^{a_n}] = (\underbrace{x_1, \dots, x_1}_{a_1}; \dots; \underbrace{x_n, \dots, x_n}_{a_n}) = \langle a_1(x_1), \dots, a_n(x_n) \rangle.$$

1.2 Операции над множествами

Обычно рассматриваются следующие операции над множествами:

1.2.1 Объединение

$$A \cup B = \{x \mid x \in A \vee x \in B\};$$

Объединение двух множеств включает все элементы, которые есть в одном из них или в обоих.

1.2.2 Пересечение

$$A \cap B = \{x \mid x \in A \wedge x \in B\};$$

Пересечение множеств включает только элементы, которые присутствуют в обоих множествах.

1.2.3 Разность

$$A \setminus B = \{x \mid x \in A \wedge x \notin B\};$$

Разность двух множеств A и B содержит все элементы, которые есть в A , но отсутствуют в B .

1.2.4 Симметрическая разность

$$A \Delta B = (A \cup B) \setminus (A \cap B) = \{x \mid (x \in A \wedge x \notin B) \vee (x \notin A \wedge x \in B)\};$$

Симметрическая разность включает элементы, которые присутствуют в одном из множеств, но не в обоих.

1.2.5 Дополнение

$$\overline{A} = \{x \mid x \notin A\}.$$

Дополнение A включает все элементы универсального множества, которые не входят в множество A .

Операция дополнения подразумевает, что задан некоторый универсум U : $\overline{A} = U \setminus A$.

1.3 Операции над мультимножествами

1.3.1 Объединение

Вхождение каждого элемента в получившемся мультимножестве C является максимумом вхождений соответствующих элементов в мультимножествах A и B .

$$C = A \cup B = \{\max(a_i(x_i), b_i(x_i))\}, \quad a_i(x_i) \in A, b_i(x_i) \in B$$

1.3.2 Пересечение

Вхождение каждого элемента в получившемся мультимножестве C является минимумом вхождений соответствующих элементов в мультимножествах A и B .

$$C = A \cap B = \{\min(a_i(x_i), b_i(x_i))\}, \quad a_i(x_i) \in A, b_i(x_i) \in B$$

1.3.3 Разность

$$C = A \setminus B = \{\min(a_i(x_i), \max(0, u_i(x_i) - b_i(x_i)))\}, a_i(x_i) \in A, b_i(x_i) \in B, u_i(x_i) \in U$$

где U — универсальное множество.

1.3.4 Симметрическая разность

$$C = A \Delta B = (A \cup B) \setminus (A \cap B) = \{\min(\max(a_i(x_i), b_i(x_i)), \max(0, u_i(x_i) - \min(a_i(x_i), b_i(x_i))))\}, a_i(x_i) \in A, b_i(x_i) \in B, u_i(x_i) \in U$$

где U — универсальное множество.

1.3.5 Дополнение

$$C = U \setminus A = \{\max(a_i x_i - u_i x_i, 0)\}, a_i x_i \in A, u_i x_i \in U$$

где U — универсальное множество.

1.3.6 Арифметическая сумма

Вхождение каждого элемента равно сумме вхождений соответствующих элементов в мультимножествах A и B , не превышающей вхождения соответствующего элемента в универсум.

$$C = A + B = \{\min(a_i x_i + b_i x_i, u_i x_i)\}, a_i x_i \in A, b_i x_i \in B, u_i x_i \in U$$

1.3.7 Арифметическая разность

Вхождение каждого элемента в мультимножество C равно разности вхождения соответствующих элементов в мультимножествах A и B .

$$C = A - B = \{\max(a_i x_i - b_i x_i, 0)\}, a_i x_i \in A, b_i x_i \in B$$

1.3.8 Арифметическое произведение

Вхождение каждого элемента в мультимножество C равно произведению вхождения соответствующих элементов в мультимножествах A и B .

$$C = A * B = \{\min(a_i x_i * b_i x_i, u_i x_i)\}, a_i x_i \in A, b_i x_i \in B, u_i x_i \in U$$

1.3.9 Арифметическое деление

Вхождение каждого элемента в мультимножество C равно частному вхождения соответствующих элементов в мультимножествах A и B .

$$C = A / B = \{\max(a_i x_i / b_i x_i, 0)\}, a_i x_i \in A, b_i x_i \in B$$

1.4 Построение бинарного кода Грея

Алгоритм генерации кода Грея создает последовательность двоичных чисел, в которой каждое последующее число отличается от предыдущего только одним битом. Для заданного количества n битов он перебирает все возможные значения от 0 до $2^n - 1$, извлекая каждый бит и формируя соответствующий код Грея с помощью побитовых операций.

Algorithm 1 Генерация кода Грея

```
1: grayCodes  $\leftarrow$  пустой список
2: for  $i \leftarrow 0$  to  $2^n - 1$  do
3:   gray  $\leftarrow$  список длины  $n$  заполненный 0
4:   for  $j \leftarrow 0$  to  $n - 1$  do
5:     gray[j]  $\leftarrow (i \gg (n - 1 - j)) \& 1$ 
6:   end for
7:   Добавить gray в grayCodes
8: end for
```

2 Особенности реализации

Программа реализует генерацию бинарного кода Грея и работу с мультимножествами заданной разрядности. В качестве основного класса используется `GrayCodeMultiset`, который отвечает за создание и управление кодами Грея, а также операциями над мультимножествами. Ключевыми особенностями реализации являются:

2.1 Структуры данных класса

Класс `GrayCodeMultiset` включает следующие структуры данных:

- `vector<vector<int>> grayCodes`: Хранит сгенерированные коды Грея.
- `vector<int> multiset1, vector<int> multiset2`: Хранение значений двух мультимножеств.
- `vector<int> universeCounts`: Хранит количество доступных элементов в универсуме для генерации мультимножеств.
- `int gray`: Хранит текущее значение разрядности для генерации кодов Грея.

2.2 Функции класса

2.2.1 Генерация кода Грея

`generateGrayCode(int n)`: Генерирует коды Грея для заданного числа разрядов n .

Вход:

- n — число разрядов (целое число).

Выход:

- `grayCodes` — вектор, содержащий коды Грея, каждый из которых представлен вектором целых чисел. Код представлен в «Листинг 1».

Листинг 1. `generateGrayCode`

```
void generateGrayCode(int n) {
    gray = n;
    grayCodes.clear();
    for (int i = 0; i < (1 << n); ++i) {
        vector<int> gray(n);
        for (int j = 0; j < n; ++j) {
            gray[j] = (i >> (n - 1 - j)) & 1;
        }
        grayCodes.push_back(gray);
    }
}
```

2.2.2 Установка количества элементов универсума

`setUniverseCounts(const vector<int> counts)`: Устанавливает количество доступных элементов в универсумах. Код представлен в «Листинг 2».

Вход:

- `counts` — вектор целых чисел, представляющий количество элементов в универсумах.

Выход:

- Прямое присвоение значений в `universeCounts` при совпадении размеров.
- В противном случае вывод сообщения об ошибке.

Листинг 2. `setUniverseCounts`

```
void setUniverseCounts(const vector<int>& counts) {
    if (counts.size() == universeCounts.size()) {
        universeCounts = counts;
    } else {
        cout << "      :  " << endl;
    }
}
```

2.2.3 Заполнение мультимножеств

`createMultisetFromUniverse(int setNumber, int n)`: Функция заполняет заданное мультимножество. Код представлен в «Листинг 3».

Вход:

- *setNumber* — номер множества для заполнения (1 или 2).
- *n* — количество элементов для заполнения.

Выход:

- Обновление `multiset1` или `multiset2` в зависимости от выбранного номера.

Листинг 3. `createMultisetFromUniverse`

```
void createMultisetFromUniverse(int setNumber, int n) {
    int pwrr = 0;
    for (size_t i = 0; i < universeCounts.size() && pwrr < n; ++i) {
        if (universeCounts[i] > 0) {
            int maxCount = std::min(universeCounts[i], n - pwrr);
            int count = (maxCount > 0) ? rand() % (maxCount + 1) : 0;
            if (setNumber == 1) {
                multiset1[i] = count;
            } else {
                multiset2[i] = count;
            }
            pwrr += count;
        }
    }
}
```

2.2.4 Объединение мультимножеств

`vector<int> unionSets() const`: Функция вычисляет объединение двух мультимножеств, возвращая вектор, где каждый элемент соответствует наибольшему количеству вхождений элемента из двух множеств. Код представлен в «Листинг 4».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий объединение двух мультимножеств.

Листинг 4. `unionSets`

```
vector<int> unionSets() const {
    vector<int> result(multiset1.size(), 0);
    for (size_t i = 0; i < multiset1.size(); ++i) {
        result[i] = max(multiset1[i], multiset2[i]); //
    }
    return result;
}
```

2.2.5 Пересечение мультимножеств

`vector<int> intersectionSets() const`: Функция вычисляет пересечение двух мультимножеств. Для каждого элемента вернёт количество вхождений, которое минимально среди двух множеств. Код представлен в «Листинг 5».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий пересечение двух мультимножеств.

Листинг 5. `intersectionSets`

```
vector<int> intersectionSets() const {
    vector<int> result(multiset1.size(), 0);
    for (size_t i = 0; i < multiset1.size(); ++i) {
        result[i] = min(multiset1[i], multiset2[i]); //
    }
    return result;
}
```

2.2.6 Разность $A \setminus B$

`vector<int> differenceSets1() const`: Функция вычисляет разность между первым множеством и вторым ($A \setminus B$), возвращая вектор, где каждый элемент составляет количество вхождений из первого множества, за вычетом тех, которые есть во втором. Код представлен в «Листинг 6».

Вход:

- Векторы целых чисел multiset1 и multiset2.

Выход:

- Вектор целых чисел, представляющий разность $A \setminus B$.

Листинг 6. differenceSets1

```
vector<int> differenceSets1() const {
    vector<int> st = complementSets2();
    vector<int> result(multiset1.size(), 0);
    for (size_t i = 0; i < multiset1.size(); ++i) {
        result[i] = min(multiset1[i], st[i]); // A \ B
    }
    return result;
}
```

2.2.7 Разность $B \setminus A$

vector<int> differenceSets2() const: Функция вычисляет разность между вторым множеством и первым ($B \setminus A$), возвращая вектор аналогично функции differenceSets1, но для второго множества. Код представлен в «Листинг 7».

Вход:

- Векторы целых чисел multiset1 и multiset2.

Выход:

- Вектор целых чисел, представляющий разность $B \setminus A$.

Листинг 7. differenceSets2

```
vector<int> differenceSets2() const {
    vector<int> st = complementSets1();
    vector<int> result(multiset1.size(), 0);
    for (size_t i = 0; i < multiset1.size(); ++i) {
        result[i] = min(st[i], multiset2[i]); // B \ A
    }
    return result;
}
```

2.2.8 Симметрическая разность

`vector<int> symmetricDifference() const`: Функция вычисляет симметрическую разность между двумя мультимножествами, возвращая элементы, которые есть только в одном из множеств, но не в обоих. Код представлен в «Листинг 8».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий симметрическую разность.

Листинг 8. `symmetricDifference`

```
vector<int> symmetricDifference() const {
    vector<int> result(multiset1.size(), 0);
    for (size_t i = 0; i < multiset1.size(); ++i) {
        result[i] = min(max(multiset1[i], multiset2[i]),
                        max(0, universeCounts[i] - min(multiset1[i],
                                                         multiset2[i])))); //
    }
    return result;
}
```

2.2.9 Дополнение A

`vector<int> complementSets1() const`

Функция вычисляет дополнение первого мультимножества относительно универсума, возвращая количество элементов, которые отсутствуют в первом множестве. Код представлен в «Листинг 9».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий количество элементов, которых нет в первом мультимножестве.

Листинг 9. `complementSets1`

```
vector<int> complementSets1() const {
    vector<int> result(multiset1.size(), 0);
    for (size_t i = 0; i < multiset1.size(); ++i) {
        result[i] = universeCounts[i] - multiset1[i]; // A
        result[i] = max(result[i], 0);
    }
    return result;
}
```

2.2.10 Дополнение В

```
vector<int> complementSets2() const
```

Функция аналогична `complementSets1`, но для второго мультимножества. Она возвращает количество элементов, которые отсутствуют во втором множестве. Код представлен в «Листинг 10».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий количество элементов, которых нет во втором мультимножестве.

Листинг 10. `complementSets2`

```
vector<int> complementSets2() const {  
    vector<int> result(multiset2.size(), 0);  
    for (size_t i = 0; i < multiset2.size(); ++i) {  
        result[i] = universeCounts[i] - multiset2[i]; //      В  
        result[i] = max(result[i], 0);  
    }  
    return result;  
}
```

2.2.11 Арифметическая сумма

```
vector<int> arithmeticSum() const
```

Функция вычисляет арифметическую сумму двух мультимножеств, возвращая вектор, где каждый элемент равен минимуму между суммой двух соответствующих элементов и максимальным количеством вхождений из универсума. Код представлен в «Листинг 11».

Вход:

- Векторы целых чисел multiset1 и multiset2.

Выход:

- Вектор целых чисел, представляющий арифметическую сумму мультимножеств.

Листинг 11. arithmeticSum

```
vector<int> arithmeticSum() const {  
    vector<int> result(multiset1.size(), 0);  
    for (size_t i = 0; i < multiset1.size(); ++i) {  
        result[i] = min(multiset1[i] + multiset2[i], universeCounts[i]); //  
    }  
    return result;  
}
```

2.2.12 Арифметическая разность A - B

```
vector<int> arithmeticDifference() const
```

Функция вычисляет разность между первым и вторым мультимножеством, возвращая вектор, где каждый элемент представляет собой разность A - B. Если разность < 0, то элемент равен 0. Код представлен в «Листинг 12».

Вход:

- Векторы целых чисел multiset1 и multiset2.

Выход:

- Вектор целых чисел, представляющий разность A - B.

Листинг 12. arithmeticDifference

```
vector<int> arithmeticDifference() const {  
    vector<int> result(multiset1.size(), 0);  
    for (size_t i = 0; i < multiset1.size(); ++i) {  
        result[i] = max(0, (multiset1[i] - multiset2[i])); //  
    }  
    return result;  
}
```

2.2.13 Арифметическая разность B - A

```
vector<int> arithmeticDifference2() const
```

Функция аналогична `arithmeticDifference`, но вычисляет разность между вторым и первым множеством, возвращая вектор, где каждый элемент представляет собой $B - A$. Код представлен в «Листинг 13».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий разность $B - A$.

Листинг 13. `arithmeticDifference2`

```
vector<int> arithmeticDifference2() const {  
    vector<int> result(multiset1.size(), 0);  
    for (size_t i = 0; i < multiset1.size(); ++i) {  
        result[i] = max(0, (multiset2[i] - multiset1[i]));  
    }  
    return result;  
}
```

2.2.14 Арифметическое произведение

```
vector<int> arithmeticProduct() const
```

Функция вычисляет арифметическое произведение элементов двух множеств, возвращая вектор, где каждый элемент является произведением соответствующих элементов из двух множеств, ограниченным универсумом. Код представлен в «Листинг 14».

Вход:

- Векторы целых чисел `multiset1` и `multiset2`.

Выход:

- Вектор целых чисел, представляющий арифметическое произведение элемента из двух множеств.

Листинг 14. `arithmeticProduct`

```
vector<int> arithmeticProduct() const {  
    vector<int> result(multiset1.size(), 0);  
    for (size_t i = 0; i < multiset1.size(); ++i) {  
        result[i] = min(universeCounts[i], multiset1[i] * multiset2[i]);  
    }  
    return result;  
}
```


2.2.15 Функция арифметического деления

```
vector<int> arithmeticDivision2() const
```

Функция выполняет деление элементов второго мультимножества на соответствующие элементы первого. Если элементы первого или второго мультимножества равны нулю, результат для этого элемента будет равен нулю. Код представлен в «Листинг 15».

Вход:

- Векторы целых чисел multiset1 и multiset2.

Выход:

- Вектор целых чисел, представляющий результат деления элементов второго мультимножества на соответствующие элементы первого.

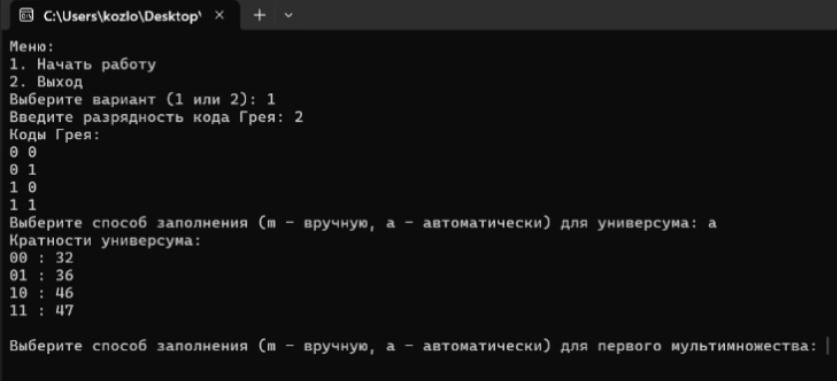
Листинг 15. arithmeticDivision2

```
vector<int> arithmeticDivision2() const {  
    vector<int> result(multiset2.size(), 0);  
    for (size_t i = 0; i < multiset2.size(); ++i) {  
        if (multiset1[i] != 0 && multiset2[i] != 0) {  
            result[i] = multiset2[i] / multiset1[i];  
        } else {  
            result[i] = 0;  
        }  
    }  
    return result;  
}
```

3 Результаты работы

3.1 Ввод универсума

Пользователь может заполнить универсум вручную или автоматически (см. рис. 1).

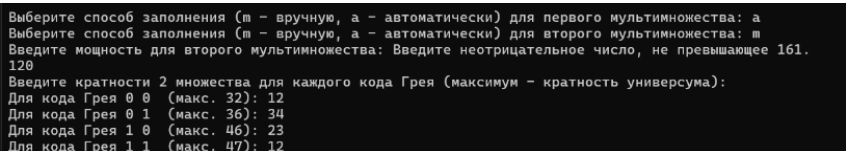


```
C:\Users\kozlo\Desktop\ x + v
Меню:
1. Начать работу
2. Выход
Выберите вариант (1 или 2): 1
Введите разрядность кода Грея: 2
Коды Грея:
0 0
0 1
1 0
1 1
Выберите способ заполнения (м - вручную, а - автоматически) для универсума: а
Кратности универсума:
00 : 32
01 : 36
10 : 46
11 : 47
Выберите способ заполнения (м - вручную, а - автоматически) для первого мультимножества: |
```

Рис. 1. Ввод универсума

3.2 Ввод множеств

Пользователь может заполнить кратности множеств вручную или автоматически (см. рис. 2).



```
Выберите способ заполнения (м - вручную, а - автоматически) для первого мультимножества: а
Выберите способ заполнения (м - вручную, а - автоматически) для второго мультимножества: м
Введите мощность для второго мультимножества: Введите неотрицательное число, не превышающее 161.
120
Введите кратности 2 множества для каждого кода Грея (максимум - кратность универсума):
Для кода Грея 0 0 (макс. 32): 12
Для кода Грея 0 1 (макс. 36): 34
Для кода Грея 1 0 (макс. 46): 23
Для кода Грея 1 1 (макс. 47): 12
```

Рис. 2. Ввод множеств

3.3 Вывод результатов

Пользователю выводятся результаты операций над мультимножествами (см. рис. 3, 4, 5).

```
Кратности универсума:  
00 : 32  
01 : 36  
10 : 46  
11 : 47  
  
Мощность универсума:  
161  
Первое мультимножество:  
00 : 32  
01 : 31  
10 : 22  
11 : 35  
Мощность 1 мультимножества:  
120  
Второе мультимножество:  
00 : 12  
01 : 34  
10 : 23  
11 : 12  
Мощность 2 мультимножества:  
81  
Объединение:  
00 : 32  
01 : 34  
10 : 23  
11 : 35
```

Рис. 3. Вывод результатов 1

```
Пересечение :
00 : 12
01 : 31
10 : 22
11 : 12

Разность A / B:
00 : 20
01 : 2
10 : 22
11 : 35

Разность B / A:
00 : 0
01 : 5
10 : 23
11 : 12

Симметрическая разность:
00 : 20
01 : 5
10 : 23
11 : 35

Дополнение первого множества:
00 : 0
01 : 5
10 : 24
11 : 12
```

Рис. 4. Вывод результатов 2

Арифметическая сумма:

00 : 32

01 : 36

10 : 45

11 : 47

Арифметическая разность(A-B):

00 : 20

01 : 0

10 : 0

11 : 23

Арифметическая разность(B-A):

00 : 0

01 : 3

10 : 1

11 : 0

Арифметическое деление (A/B)

00 : 2

01 : 0

10 : 0

11 : 2

Рис. 5. Вывод результатов 3

3.4 Некорректный ввод

При попытке некорректного ввода пользователю выдётся предупреждение об ошибке и дается возможность повторного ввода (см. рис. 6).

```
Выберите способ заполнения (m – вручную, a – автоматически) для универсума: 2
Ошибка: введите 'm' или 'a'.
```

Рис. 6. Некорректный ввод

3.5 Ввод пустого универсума

Пользователь может вводить пустой универсум (см. рис. 7).

```
Меню:
1. Начать работу
2. Выход
Выберите вариант (1 или 2): 1
Введите разрядность кода Грея: 0
Универсум – пустое множество
Мощность универсума: 0
Первое мультимножество: пустое множество
Мощность 1 мультимножества: 0
Второе мультимножество: пустое множество
Мощность 2 мультимножества: 0
Объединение: пустое множество
Пересечение: пустое множество
Разность A / B: пустое множество
Разность B / A: пустое множество
Симметрическая разность: пустое множество
Дополнение первого множества: пустое множество
Дополнение второго множества: пустое множество
Арифметическая сумма: пустое множество
Арифметическое умножение: пустое множество
Арифметическая разность A – B: пустое множество
Арифметическая разность B – A: пустое множество
Результаты деления A/B: пустое множество
Результаты деления B/A: пустое множество
Меню:
1. Начать работу
2. Выход
Выберите вариант (1 или 2): |
```

Рис. 7. Ввод пустого универсума

Заключение

В данной лабораторной работе была разработана программа для генерации бинарного кода Грея и выполнения операций над мультимножествами.

В программе реализованы следующие операции над мультимножествами:

1. Объединение множеств

- Функция: `unionSets()`

2. Пересечение множеств

- Функция: `intersectionSets()`

3. Разность множеств

- Функции: `differenceSets1()` и `differenceSets2()`

4. Симметрическая разность

- Функция: `symmetricDifference()`

5. Дополнение множеств

- Функции: `complementSets1()` и `complementSets2()`

6. Арифметическая сумма

- Функция: `arithmeticSum()`

7. Арифметическое умножение

- Функция: `arithmeticProduct()`

8. Арифметическая разность

- Функции: `arithmeticDifference()` и `arithmeticDifference2()`

9. Арифметическое деление

- Функции: `arithmeticDivision()` и `arithmeticDivision2()`

Преимущества программы:

- Для универсума есть возможность как ручного, так и автоматического ввода.
- Есть возможность заново запустить программу после вывода всех операций.
- Четкое разделение функций операций, что позволяет использовать их при вычислении других операций(например в функции разности используется функция дополнения).

Недостатки программы:

- Избыточное количество функций, предназначенных для выполнения аналогичных операций (разность, арифметическая разность, деление).
- отсутствие отдельной функции для проверки корректности ввода данных. В текущем варианте программа выполняет проверку ввода непосредственно в функции `main`.

Масштабируемость:

Программа может быть легко расширена для включения дополнительных операций и функций, таких как:

- Разработать графический интерфейс для облегчения взаимодействия пользователя с программой.
- Дать пользователю возможность выбирать, результаты каких конкретных операций он хочет увидеть.
- Добавить возможность работы с 3 и более множествами.

Список использованной литературы

- [1] Новиков, Ф.А. *Н73 Дискретная математика для программистов: Учебник для вузов. 3-е изд.* — СПб.: Питер, 2009. — 384 с.: ил. — (Серия «Учебник для вузов»)).