

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
02.03.01 Математика и компьютерные науки

Отчёт по дисциплине

Дискретная математика
Лабораторная работа №1

Выполнил:

студент _____

Принял:

«_____» _____ 2025 г.

Санкт-Петербург, 2025

Содержание

Введение	3
1 Постановка задачи	4
2 Математическое описание	5
3 Особенности реализации	8
3.1 Класс Multiset	8
3.2 Класс Interface	22
4 Результаты работы программы	38
Заключение	48
Список литературы	51
Приложение 1. Файл Multiset.h	52
Приложение 2. Файл Multiset.cpp	54
Приложение 3. Файл Interface.h	64
Приложение 4. Файл Interface.cpp	65
Приложение 5. Файл main.cpp	79

Введение

Актуальность и цель работы

Использование кода Грея позволяет минимизировать количество ошибок при переходе от одного состояния к другому, что применяется, например, в кодировании данных. Мультимножества позволяют описывать системы, где элемент может встречаться несколько раз — это востребовано в базах данных, статистике и других областях.

Таким образом, разработка программного инструмента, который демонстрирует генерацию кода Грея и операции над мультимножествами, способствует формированию навыков работы с дискретными структурами данных и алгоритмами, а также помогает на практике закрепить фундаментальные понятия теории множеств.

1 Постановка задачи

В рамках лабораторной работы необходимо разработать программное обеспечение, которое продемонстрирует на практике работу с бинарным кодом Грея и мультимножествами. Программа должна быть интерактивной, устойчивой к ошибкам пользователя и обеспечивать понятный интерфейс для формирования и обработки множеств.

Цель данной работы — реализовать программу, которая:

- генерирует бинарный код Грея заданной разрядности;
- формирует мультимножества двумя способами — ручным и автоматическим;
- выполняет над ними стандартные и арифметические операции;
- защищает пользователя от некорректного ввода;
- позволяет создавать много мультимножеств и выбирать из них те, над которыми будут проводиться операции.

2 Математическое описание

Код Грея. Бинарный код Грея — это последовательность n -битных двоичных слов, в которой любые два соседних слова отличаются ровно на один бит.

Для целого i от 0 до $2^n - 1$ код Грея $G(i)$ вычисляется по формуле:

$$G(i) = i \oplus \left\lfloor \frac{i}{2} \right\rfloor,$$

где \oplus — побитовое исключающее ИЛИ (XOR).

Универсум мультимножеств. Пусть задана разрядность n и максимальная кратность m . Универсум U :

$$U = \{x_1^{m_1}, x_2^{m_2}, \dots, x_{2^n}^{m_{2^n}}\}; n \in N; m \in N \cup \{0\}$$

где x_i — n -битовый код Грея с кратностью m_i .

Мультимножества. Мультимножество — подмножество универсума, которое задается следующим способом:

$$A = \{x_1^{k_1}, x_2^{k_2}, \dots, x_{2^n}^{k_{2^n}}\}; k_i \leq m_i.$$

Операции над мультимножествами. Для любых двух мультимножеств A и B с универсумом U операции задаются следующим образом:

- **Объединение:**

$$A \cup B = \{x^{m_{AB}} \mid m_{AB} = \max(m_A, m_B)\},$$

где m_A — кратность x в A , m_B — в B .

- **Пересечение:**

$$A \cap B = \{x^{m_{AB}} \mid m_{AB} = \min(m_A, m_B)\}.$$

- **Разность:**

$$A \setminus B = \{x^{m_{AB}} \mid m_{AB} = \max(m_A - m_B, 0)\}.$$

- **Симметрическая разность:**

$$A \Delta B = (A \cup B) \setminus (A \cap B).$$

- **Дополнение:**

$$\overline{A} = \{x^{m_{AB}} \mid m_{AB} = m - m_A\},$$

где m — максимальная кратность элемента универсума.

- **Арифметическая сумма:**

$$A + B = \{x^{m_{AB}} \mid m_{AB} = \min(m_A + m_B, m)\},$$

где m - максимальная кратность элемента m_A - кратность элемента в мультимножестве A , m_B - кратность элемента в мультимножестве B .

- **Арифметическая разность:**

$$A - B = \{x^{m_{AB}} \mid m_{AB} = \max(m_A - m_B, 0)\},$$

- **Арифметическое произведение:**

$$A * B = \{x^{m_{AB}} \mid m_{AB} = \min(m_A * m_B, m)\}.$$

- **Арифметическое деление:**

$$A \div B = \{x^{m_{AB}} \mid m_{AB} = \max\left(\left\lfloor \frac{m_A}{m_B} \right\rfloor, 0\right)\}.$$

Таким образом, программа должна:

1. сгенерировать универсум U кода Грея для заданного n ;
2. формировать мультимножества (для операций необходимо хотя бы 2; универсум можно использовать);
3. проводить все операции, определённые выше.

3 Особенности реализации

Программа состоит из двух классов: `Multiset` и `Interface`.

Класс `Multiset` содержит в себе поля для хранения данных о мультимножестве и методы для проведения операций над мультимножествами. Мультимножества хранятся в контейнере `std::string`, т.к. с ним удобно работать.

Класс `Interface` содержит в себе поля для хранения данных о всех мультимножествах в контейнере `std::map<std::string, Multiset>`, где ключи - названия мультимножеств, а значения - сами мультимножества. Методы класса отвечают за консольное меню (пользовательский ввод, вывод мультимножеств и результатов операций над ними).

3.1 Класс `Multiset`

- Метод `generateGrayCode`

Вход: `int n` - разрядность кода Грея.

Выход: `std::vector<std::string> gray` - контейнер с кодами Грея.

Назначение: генерирует коды Грея разрядности n .

Код:

```
1 if (n <= 0) return {};  
2 std::vector<std::string> gray;  
3 int total = 1 << n;  
4  
5 for (int i = 0; i < total; ++i) {  
6     int grayCode = i ^ (i >> 1);  
7     std::string code = "";  
8     for (int j = n - 1; j >= 0; --j) {  
9         code += (grayCode & (1 << j)) ? '1' : '0';
```



```

10     }
11     gray.push_back(code);
12 }
13 return gray;

```

- **Метод** `createUniverse`

Вход: `int depth` - разрядность кода Грея, который будет храниться в универсуме.

Выход: `std::vector<std::string> universe` - универсум по кодам Грея с заданной разрядностью.

Назначение: создаёт универсум по кодам Грея.

Код:

```

1 void Multiset::createUniverse(int depth) {
2     universe = generateGrayCode(depth);
3 }

```

- **Метод** `setMaxMultiplicity`

Вход: `int limit` - значение максимальной кратности элементов.

Выход: `int limit` - значение максимальной кратности элементов.

Назначение: устанавливает максимальную кратность в универсуме (хранится в виде статической переменной).

Код:

```

1 void Multiset::setMaxMultiplicity(int limit) {
2     maxMultiplicity = limit;
3 }

```

- **Метод** `getMaxMultiplicity`

Вход: `int maxMultiplicity` - значение максимальной кратности элементов.

Выход: `int maxMultiplicity` - значение максимальной кратности элементов.

Назначение: возвращает текущую максимальную кратность.

Код:

```
1 int Multiset::getMaxMultiplicity() {  
2     return maxMultiplicity;  
3 }
```

- **Метод** `size`

Вход: `Multiset input` - мультимножество.

Выход: `int total` - мощность мультимножества.

Назначение: возвращает мощность мультимножества.

Код:

```
1 int Multiset::size() const {  
2     int total = 0;  
3     for (const auto& p : elements) total += p.second;  
4     return total;  
5 }
```

- **Метод** `fillManual`

Вход: `int size` - мощность создаваемого мультимножества.

Выход: `Multiset multi` - мультимножество.

Назначение: воздание и ручное заполнение мультимножества.

Код:

```
1 void Multiset::fillManual(int size) {  
2     std::cin.ignore();  
3     if (size < 0) return;  
4  
5     if (size != 0)
```

```

6      std::cout << "Enter " << size << " Gray codes (limit: " << maxMultiplicity <<
      "\n"
7      << "(You can enter either Gray code only or Gray code [space] count; other
      numbers won't be accounted)" << std::endl;
8
9      for (int i = 0; i < size; ) {
10         std::string code;
11         while (true) {
12             std::cout << "Code #" << (i + 1) << ": ";
13
14             std::string line;
15             std::getline(std::cin, line);
16             std::istringstream iss(line);
17
18             std::string code;
19             if (!(iss >> code)) {
20                 std::cout << "Empty input. Try again.\n";
21                 continue;
22             }
23
24             int count = 1;
25             iss >> count;
26
27             if (!isValidCode(code)) {
28                 std::cout << "Invalid code. Valid: ";
29                 for (const auto& c : universe) std::cout << c << " ";
30                 std::cout << "\nTry again.\n";
31                 continue;
32             }
33
34             if (!canAdd(code)) {
35                 std::cout << "You've reached your limit(" << maxMultiplicity

```

```

36         << " ) for the code \"" << code << "\". Please pick different.\n";
37         continue;
38     }
39
40     int currentCount = frequency(code);
41     if (currentCount + count > maxMultiplicity) {
42         std::cout << "Adding " << count << " of \"" << code
43         << "\" would exceed limit (" << maxMultiplicity
44         << "). Currently have " << currentCount << ".\n"
45         << "Try again with fewer or different code.\n";
46         continue;
47     }
48
49     for (int j = 0; j < count; ++j) {
50         add(code);
51     }
52
53     i += count;
54     break;
55 }
56 }
57 }

```

- Метод `fillRandom`

Вход: `int size` - мощность создаваемого мультимножества.

Выход: `Multiset multi` - мультимножество.

Назначение: создание и заполнение мультимножества случайными элементами.

Код:

```

1 void Multiset::fillRandom(int size) {
2     if (size <= 0 || universe.empty()) return;

```

```

3
4     for (int i = 0; i < size; ++i) {
5         std::string code;
6         do {
7             code = universe[rand() % universe.size()];
8         } while (!canAdd(code));
9
10        add(code);
11    }
12 }

```

- Метод `frequency`

Вход: `const std::string& code` - код Грея.

Выход: `int count` - кратность элемента в мультимножестве.

Назначение: возвращает кратность элемента (кода Грея) в мультимножестве.

Код:

```

1 int Multiset::frequency(const std::string& code) const {
2     auto it = elements.find(code);
3     return (it != elements.end()) ? it->second : 0;
4 }

```

- Метод `canAdd`

Вход: `const std::string& code` - код Грея.

Выход: `bool canAdd` - возможность добавления элемента в мультимножество.

Назначение: проверяет, можно ли добавить элемент (код Грея) в мультимножество.

Код:

```

1 bool Multiset::canAdd(const std::string& code) const {
2     // Check if code is valid and if adding one more would exceed maxMultiplicity
3     if (!isValidCode(code)) return false;
4     int current = frequency(code);
5     return current < maxMultiplicity;
6 }

```

- **Метод** `isValidCode`

Вход: `const std::string& code` - код Грея.

Выход: `bool isValidCode` - корректность данного кода Грея (есть ли он в универсуме).

Назначение: проверяет корректность данного кода Грея (принадлежит ли код универсуму).

Код:

```

1 bool Multiset::isValidCode(const std::string& code) const {
2     for (const auto& u : universe) {
3         if (code == u) return true;
4     }
5     return false;
6 }

```

- **Метод** `add`

Вход: `const std::string& code, int count = 1` - код, который нужно добавить, и его кратность (по умолч. 1).

Выход: `int added` - кол-во реально добавленных элементов.

Назначение: добавляет элементы в массив.

Код:

```

1 int Multiset::add(const std::string& code, int count = 1) {

```

```

2   if (!isValidCode(code) || count <= 0) return 0;
3
4   int current = frequency(code);
5   int available = maxMultiplicity - current;
6   int actualToAdd = std::min(count, available);
7
8   if (actualToAdd > 0) {
9       elements[code] += actualToAdd;
10  }
11
12  return actualToAdd;
13 }

```

- Метод `unionWith`

Вход: `Multiset other` - другое мультимножество.

Выход: `Multiset result` - новое мультимножество.

Назначение: объединение (максимум кратностей).

Код:

```

1 Multiset Multiset::unionWith(const Multiset& other) const {
2     Multiset result;
3
4     for (const auto& p : elements) {
5         if (p.second > 0) {
6             result.elements[p.first] = p.second;
7         }
8     }
9     for (const auto& p : other.elements) {
10        if (p.second > 0) {
11            result.elements[p.first] = std::max(result.elements[p.first], p.second);
12        }
13    }

```

```

14     return result;
15 }

```

- **Метод** `intersectionWith`

Вход: `Multiset other` - другое мультимножество.

Выход: `Multiset result` - новое мультимножество.

Назначение: пересечение (минимум кратностей).

Код:

```

1 Multiset Multiset::intersectionWith(const Multiset& other) const {
2     Multiset result;
3
4     for (const auto& p : elements) {
5         if (other.elements.count(p.first)) {
6             result.elements[p.first] = std::min(p.second, other.elements.at(p.first));
7         }
8     }
9     return result;
10 }

```

- **Метод** `differenceWith`

Вход: `Multiset other` - другое мультимножество.

Выход: `Multiset result` - новое мультимножество.

Назначение: разность A B.

Код:

```

1 Multiset Multiset::differenceWith(const Multiset& other) const {
2     return intersectionWith(other.complement());
3 }

```

- **Метод** `symmetricDifferenceWith`

Вход: `Multiset other` - другое мультимножество.

Выход: Multiset result новое мультимножество.

Назначение: симметрическая разность.

Код:

```
1 Multiset Multiset::symmetricDifferenceWith(const Multiset& other) const {  
2     Multiset result;  
3     auto uni = unionWith(other);  
4     auto inter = intersectionWith(other);  
5     result = uni.differenceWith(inter);  
6  
7     return result;  
8 }
```

- **Метод** arithmeticSum

Вход: Multiset other - другое мультимножество.

Выход: Multiset result - новое мультимножество.

Назначение: арифметическая сумма.

Код:

```
1 Multiset Multiset::arithmeticSum(const Multiset& other) const {  
2     Multiset result;  
3  
4     for (const auto& p : elements) result.elements[p.first] += p.second;  
5     for (const auto& p : other.elements) result.elements[p.first] += p.second;  
6     for (auto& p : result.elements)  
7         if (p.second > maxMultiplicity)  
8             p.second = maxMultiplicity;  
9  
10    return result;  
11 }
```

- **Метод** arithmeticDifferenceWith

Вход: `Multiset other` - другое мультимножество.

Выход: `Multiset result` - новое мультимножество.

Назначение: арифметическая разность.

Код:

```
1 Multiset Multiset::arithmeticDifferenceWith(const Multiset& other) const {
2     Multiset result;
3
4     for (const auto& p : elements) {
5         int count = p.second;
6         if (other.elements.count(p.first)) {
7             count -= other.elements.at(p.first);
8         }
9         if (count > 0) {
10             result.elements[p.first] = count;
11         }
12     }
13     return result;
14 }
```

- **Метод** `arithmeticProduct`

Вход: `Multiset other` - другое мультимножество.

Выход: `Multiset result` - новое мультимножество.

Назначение: арифметическое произведение.

Код:

```
1 Multiset Multiset::arithmeticProduct(const Multiset& other) const {
2     Multiset result;
3
4     for (const auto& p : elements) {
5         if (other.elements.count(p.first)) {
6             result.elements[p.first] = std::min(p.second * other.elements.at(p.first),
```

```

7         }
8     }
9
10    return result;
11 }

```

- **Метод** `arithmeticDivision`

Вход: `Multiset other` - другое мультимножество.

Выход: `Multiset result` - новое мультимножество.

Назначение: арифметическое деление.

Код:

```

1 Multiset Multiset::arithmeticDivision(const Multiset& other) const {
2     Multiset result;
3
4     for (const auto& p : elements)
5         if (other.elements.count(p.first) && other.elements.at(p.first) > 0)
6             if (p.second / other.elements.at(p.first) > 0)
7                 result.elements[p.first] = p.second / other.elements.at(p.first);
8
9     return result;
10 }

```

- **Метод** `complement`

Вход: `Multiset current` - текущее мультимножество.

Выход: `Multiset result` - новое мультимножество, являющееся дополнением текущего.

Назначение: дополнение относительно максимальной кратности.

Код:

```

1 Multiset Multiset::complement() const {
2     Multiset result;
3
4     for (const auto& code : universe) {
5         int current = frequency(code);
6         int needed = maxMultiplicity - current;
7         if (needed > 0) {
8             result.elements[code] = needed;
9         }
10    }
11
12    return result;
13 }

```

- Метод `print`

Вход: `Multiset multi` - текущее мультимножество.

Выход: `Multiset multi` - текущее мультимножество.

Назначение: печатают мультимножество или универсум.

Код:

```

1 void Multiset::print() const {
2     std::cout << "{ ";
3     bool first = true;
4     for (const auto& p : elements) {
5         if (!first) std::cout << ", ";
6         std::cout << "\"\" << p.first << "\"^\" << p.second;
7         first = false;
8     }
9     std::cout << "}" << std::endl;
10 }

```

- Метод `printU`

Вход: `std::vector<std::string> universe` - универсум.

Выход: `std::vector<std::string> universe` - универсум.

Назначение: печатает универсум.

Код:

```
1 void Multiset::printU() const {
2     std::cout << "{ ";
3     bool first = true;
4     for (const auto& p : universe) {
5         if (!first) std::cout << ", ";
6         std::cout << "\"" << p << "\"^" << 10;
7         first = false;
8     }
9     std::cout << "}" << std::endl;
10 }
```

- **Метод** `getUniverse`

Вход: `std::vector<std::string> universe` - универсум.

Выход: `Multiset multiUni` универсум, представленный в виде типа `Multiset`.

Назначение: возвращает универсум в виде типа `Multiset`.

Код:

```
1 Multiset Multiset::getUniverse() {
2     Multiset uni;
3     for (const auto& elem : universe) {
4         uni.elements[elem] = maxMultiplicity;
5     }
6     return uni;
7 }
```

3.2 Класс Interface

- Метод isValidName

Вход: `const std::string& name` - имя нового мультимножества.

Выход: `bool correct` - валидно ли имя мультимножества.

Назначение: проверяет, валидно ли имя мультимножества.

Код:

```
1 bool Interface::isValidName(const std::string& name) {
2     bool isValid = data_.find(name) == data_.end() && name != "";
3     if (isValid)
4         return true;
5     else
6         std::cout << "Multiset with this name already exists or name is invalid.\n"
7         << "Please, try again: ";
8     return false;
9 }
```

- Метод createMultisets

Вход: `bool rand` - режим создания мультимножества (ручное или случайное).

Выход: `Multiset multi` - новое мультимножество.

Назначение: создаёт новое мультимножество ручным или случайным способом; если нет универсума, сначала создает его.

Код:

```
1 void Interface::createMultisets(bool rand) {
2     if (data_.size() == 0) { // Here we set the universe parameters
3         std::cout << "Enter bit depth: ";
4         int depth;
5         while (!(std::cin >> depth) || depth < 0) {
6             std::cout << "Invalid input. Enter non-negative integer: ";
```

```

7         IGN
8     }
9
10    bitDepth_ = depth;
11
12    if (bitDepth_ == 0)
13        goto AddUniverse;
14
15    Multiset::createUniverse(depth);
16
17    std::cout << "Enter universum multiplicity: ";
18    int multi;
19    IGN
20    while (!(std::cin >> multi) || multi < 0) {
21        std::cout << "Invalid input. Enter non-negative integer: ";
22        IGN
23    }
24    Multiset::setMaxMultiplicity(multi);
25    uniMultiplicity_ = multi;
26
27    AddUniverse:
28    if (uniMultiplicity_ == 0)
29        Multiset::createUniverse(0);
30    data_["U"] = Multiset::getUniverse();
31    uniSize_ = Multiset::getUniverse().size();
32 }
33
34 if (rand) {
35     createRandom();
36 }
37 else {
38     Multiset A;

```

```

39
40     std::cout << "Enter name of the multiset: ";
41     std::string name;
42     std::cin >> name;
43     while (!isValidName(name)) {
44         IGN
45         std::cin >> name;
46     }
47     if (bitDepth_ * uniMultiplicity_ == 0) {
48         goto AddMultiset;
49     }
50     std::cout << "\nHow to fill multiset?\n1. Manual\n2. Auto\n> ";
51     int choice;
52     IGN
53     while (!(std::cin >> choice) || (choice != 1 && choice != 2)) {
54         IGN
55         std::cout << "Enter 1 or 2: ";
56     }
57
58     int sizeA;
59     std::cout << "Enter size of the multiset " << name << ": ";
60     IGN
61     while (!(std::cin >> sizeA) || sizeA < 0 || sizeA > uniSize_) {
62         IGN
63         std::cout << "Size can't be negative or more than universe's size: ";
64     }
65
66     if (choice == 1) {
67         std::cout << "Fill Multiset " << name << ':' << std::endl;
68         A.fillManual(sizeA);
69     }
70     else {

```



```

71         A.fillRandom(sizeA);
72     }
73     AddMultiset:
74     data_[name] = A;
75 }
76 }

```

- **Метод** `displayMenu`

Вход: `std::string` - текст, который нужно вывести в пользовательском меню.

Выход: `std::string` - текст, который нужно вывести в пользовательском меню.

Назначение: выводит меню программы.

Код:

```

1 void Interface::displayMenu() {
2     std::cout << "Choose an option:\n"
3     << "1. Create new multisets\n"
4     << "2. Perform an operation on existing multisets\n"
5     << "3. Print a multiset\n"
6     << "4. Print all multisets' names\n"
7     << "5. Print all multisets\n"
8     << "6. Create a random multiset\n"
9     << "7. Delete a multiset\n"
10    << "8. Reset (delete all multisets and universe)\n"
11    << "0. Exit\n> ";
12 }

```

- **Метод** `performAll`

Вход: `Multiset multi1, multi2; std::string nameA, nameB` - мультимножества и их имена.

Выход: `Multiset` - результат выполнения операций над мультимножествами.

Назначение: выполняет все операции и выводит результаты.

Код:

```
1 void Interface::performAll(Multiset& A, Multiset& B, std::string& nameA, std::string&
   nameB) {
2     std::cout << "\n===== MULTISSETS =====\n";
3     std::cout << "U: "; A.printU(); std::cout << '\n';
4     std::cout << nameA << ": "; A.print(); std::cout << '\n';
5     std::cout << nameB << ": "; B.print(); std::cout << '\n';
6     std::cout << "\n===== OPERATIONS =====\n";
7     std::cout << nameA << " union " << nameB << ": "; A.unionWith(B).print();
   std::cout << '\n';
8     std::cout << nameA << " inter " << nameB << ": "; A.intersectionWith(B).
   print(); std::cout << '\n';
9     std::cout << nameA << " \ " << nameB << ": "; A.differenceWith(B).print();
   std::cout << '\n';
10    std::cout << nameB << " \ " << nameA << ": "; B.differenceWith(A).print();
   std::cout << '\n';
11    std::cout << nameA << " /\ " << nameB << ": "; A.symmetricDifferenceWith
   (B).print(); std::cout << '\n';
12    std::cout << nameA << " + " << nameB << ": "; A.arithmeticSum(B).print();
   std::cout << '\n';
13    std::cout << nameA << " - " << nameB << ": "; A.arithmeticDifferenceWith(B
   ).print(); std::cout << '\n';
14    std::cout << nameB << " - " << nameA << ": "; B.arithmeticDifferenceWith(A
   ).print(); std::cout << '\n';
15    std::cout << nameA << " * " << nameB << ": "; A.arithmeticProduct(B).print
   (); std::cout << '\n';
16    std::cout << nameA << " / " << nameB << ": "; A.arithmeticDivision(B).print
   (); std::cout << '\n';
17    std::cout << nameB << " / " << nameA << ": "; B.arithmeticDivision(A).print
```

```

    ); std::cout << '\n';
18    std::cout << "Complement of " << nameA << ": "; auto compA = A.
        complement(); compA.print(); std::cout << '\n';
19    std::cout << "Complement of " << nameB << ": "; auto compB = B.
        complement(); compB.print(); std::cout << '\n';
20 }

```

- **Метод** `perform`

Вход: `std::string nameA, nameB` - имена мультимножеств.

Выход: `Multiset` - результат выполнения операций над мультимножествами.

Назначение: позволяет выполнять выбранную операцию.

Код:

```

1 void Interface::perform(std::string& nameA, std::string& nameB) {
2     std::cout << "Enter the name of the 1st multiset: ";
3     std::cin >> nameA;
4     while (isValidName(nameA)) {
5         std::cout << "Multiset with this name doesn't exist. Please, try again: ";
6         std::cin >> nameA;
7     }
8     std::cout << "Enter the name of the 2nd multiset: ";
9     std::cin >> nameB;
10    while (isValidName(nameB)) {
11        std::cout << "Multiset with this name doesn't exist. Please, try again: ";
12        std::cin >> nameB;
13    }
14
15    Multiset compA;
16    Multiset compB;
17    int trash;
18    int choice;

```

```

19
20 while (true) {
21     CL;
22     std::cout << "Choose an operation:\n"
23     << "1. ' union '" << nameA << "' union '" << nameB << "'\n"
24     << "2. ' inter '" << nameB << "' inter '" << nameB << "'\n"
25     << "3. ' \ \ '" << nameA << "' \ \ '" << nameB << "'\n"
26     << "4. ' \ \ '" << nameB << "' \ \ '" << nameA << "'\n"
27     << "5. ' /\ \ '" << nameA << "' /\ \ '" << nameB << "'\n"
28     << "6. ' + '" << nameA << "' + '" << nameB << "'\n"
29     << "7. ' - '" << nameA << "' - '" << nameB << "'\n"
30     << "8. ' * '" << nameA << "' * '" << nameB << "'\n"
31     << "9. ' / '" << nameA << "' / '" << nameB << "'\n"
32     << "0. ' / '" << nameB << "' / '" << nameA << "'\n"
33     << "d. Do all operations\n"
34     << "a. Complement of '" << nameA << "'\n"
35     << "b. Complement of '" << nameB << "'\n"
36     << "q. Go to main menu\n>";
37
38     choice = _getch();
39
40     switch (choice) {
41         case '1':
42             std::cout << "A union B: "; data_[nameA].unionWith(data_[nameB]).
43             print(); std::cout << '\n';
44             W;
45             break;
46         case '2':
47             std::cout << "A inter B: "; data_[nameA].intersectionWith(data_[nameB
48             ]).print(); std::cout << '\n';
49             W;
50             break;

```

```

49     case '3':
50         std::cout << "A \\ B: "; data_[nameA].arithmeticDifferenceWith(data_[
            nameB]).print(); std::cout << '\n';
51     W;
52     break;
53     case '4':
54         std::cout << "B \\ A: "; data_[nameB].arithmeticDifferenceWith(data_[
            nameA]).print(); std::cout << '\n';
55     W;
56     break;
57     case '5':
58         std::cout << "A sym \\ B: "; data_[nameA].symmetricDifferenceWith(
            data_[nameB]).print(); std::cout << '\n';
59     W;
60     break;
61     case '6':
62         std::cout << "A + B: "; data_[nameA].arithmeticSum(data_[nameB]).
            print(); std::cout << '\n';
63     W;
64     break;
65     case '7':
66         std::cout << "A - B: "; data_[nameA].differenceWith(data_[nameB]).
            print(); std::cout << '\n';
67     W;
68     break;
69     case '8':
70         std::cout << "A * B: "; data_[nameA].arithmeticProduct(data_[nameB]).
            print(); std::cout << '\n';
71     W;
72     break;
73     case '9':
74         std::cout << "A / B: "; data_[nameA].arithmeticDivision(data_[nameB]).

```

```

        print(); std::cout << '\n';
75     W;
76     break;
77     case '0':
78         std::cout << "B / A: "; data_[nameB].arithmeticDivision(data_[nameA]).
            print(); std::cout << '\n';
79     W;
80     break;
81     case 'd':
82         performAll(data_[nameA], data_[nameB], nameA, nameB);
83     W;
84     break;
85     case 'a':
86         std::cout << "Complement of A: "; compA = data_[nameA].complement
            (); compA.print(); std::cout << '\n';
87     W;
88     break;
89     case 'b':
90         std::cout << "Complement of B: "; compB = data_[nameB].complement
            (); compB.print(); std::cout << '\n';
91     W;
92     break;
93     case 'q':
94         goto Break;
95     default:
96         std::cout << "Invalid option. Please, try again!\n";
97     W;
98 }
99 continue;
100 Break:
101 break;
102 }

```

- Метод `printMultiset`

Вход: `std::string name` - имя мультимножества.

Выход: `Multiset multi` - мультимножество.

Назначение: выводит содержимое множества.

Код:

```
1 void Interface::printMultiset(std::string& name) {
2     data_[name].print();
3 }
```

- Метод `printNames`

Вход: `std::string` - имена мультимножеств.

Выход: `std::string` - имена мультимножеств.

Назначение: выводит список имён всех множеств.

Код:

```
1 void Interface::printNames() {
2     int i = 1;
3     for (const auto& pair : data_) {
4         std::cout << "#" << i << ": " << pair.first << std::endl;
5         i++;
6     }
7 }
```

- Метод `printAllMultisets`

Вход: `Multiset` - все созданные мультимножества.

Выход: `Multiset` - все созданные мультимножества.

Назначение: печатает все мультимножества.

Код:

```

1 void Interface::printAllMultisets() {
2     int i = 1;
3     for (const auto& pair : data_) {
4         std::cout << '#' << i << " \\" << pair.first << "\": ";
5         pair.second.print();
6         i++;
7     }
8 }

```

- **Метод** createRandom

Вход: Multiset universe - универсум, на основе которого будет создаваться мультимножество.

Выход: Multiset multi - мультимножество.

Назначение: создаёт случайное мультимножество.

Код:

```

1 void Interface::createRandom() {
2     std::string name;
3     int i = 10000;
4     do { name = std::to_string(rand() % i++); } while (!isValidName(name));
5     Multiset A;
6     i = 100;
7     int size = rand() % i;
8     while (size > uniMultiplicity_ * bitDepth_)
9         size--;
10    A.fillRandom(size);
11    data_[name] = A;
12    std::cout << "Random multiset created.\n"
13        << "Name: " << name
14        << "\nSize: " << A.size() << std::endl;
15 }

```


- **Метод** `deleteMultiset()`

Вход: `std::string name` - имя удаляемого мультимножества.

Выход: `std::map<std::string, Multiset> data` - контейнер мультимножеств (обновленный: удалено выбранное мультимножество).

Назначение: удаляет мультимножество.

Листинг:

```
1 void Interface::deleteMultiset() {
2     std::string name;
3     std::cout << "Enter the name of the multiset to delete: ";
4     std::cin >> name;
5     while (isValidName(name) || name == "U") { // Checks if it DOESN'T exist yet.
6         if (isValidName(name))
7             std::cout << "Multiset with this name doesn't exist. Please, try again: ";
8         else
9             std::cout << "Can't delete universe. Please, try again: ";
10        std::cin >> name;
11    }
12    data_.erase(name);
13    std::cout << "Deleted." << std::endl;
14 }
```

- **Метод** `reset`

Вход: `std::map<std::string, Multiset> data` - контейнер мультимножеств.

Выход: `std::map<std::string, Multiset> data` - пустой контейнер мультимножеств.

Назначение: удаляет все мультимножества.

Листинг:

```
1 void Interface::reset() {
2     data_.clear();
```

```

3     bitDepth_ = 0;
4     uniMultiplicity_ = 0;
5 }

```

- **Метод** `run`

Вход: `std::iostream` - поля ввода и вывода программы.

Выход: `int` - статус завершения работы программы.

Назначение: запускает основной цикл программы.

Листинг:

```

1  int Interface::run() {
2      int trash;
3      std::string nameA;
4      std::string nameB;
5
6      std::cout << "Hello!\n"
7      << "Here you can create multisets and perform operations upon them.\n"
8      << "Press any key to continue.";
9      trash = _getch();
10
11     while (true) {
12         CL;
13         displayMenu();
14         Again:
15         char option = _getch();
16         switch (option) {
17             case '1':
18                 CL;
19                 createMultisets();
20                 std::cout << "Press any key to continue.";
21                 trash = _getch();
22                 break;

```

```

23
24     case '2':
25         CL;
26         if (data_.size() < 2) {
27             std::cout << "You need at least two multisets to perform operations.\n"
28                 << "Press any key to continue.";
29             trash = _getch();
30             break;
31         }
32         perform(nameA, nameB);
33         break;
34
35     case '3':
36         CL;
37         if (data_.size() == 0) {
38             std::cout << "Nothing to print. No multisets available." << std::endl;
39             W;
40             break;
41         }
42         std::cout << "Enter the name of a multiset to print: ";
43         std::cin >> nameA;
44         while (isValidName(nameA)) { // Checks if it DOESN'T exist yet.
45             std::cout << "Wrong name. Please try again: ";
46             std::cin >> nameA;
47         }
48         printMultiset(nameA);
49         W;
50         break;
51
52     case '4':
53         CL;

```

```

54         if (data_.size() == 0) {
55             std::cout << "Nothing to print. No multisets available." << std::endl;
56             W;
57             break;
58         }
59         printNames();
60         W;
61         break;
62
63         case '5':
64             CL;
65             if (data_.size() == 0) {
66                 std::cout << "Nothing to print. No multisets available." << std::endl;
67                 W;
68                 break;
69             }
70             printAllMultisets();
71             W;
72             break;
73
74         case '6':
75             CL;
76             if (data_.size() == 0)
77                 createMultisets(true);
78             else
79                 createRandom();
80             W;
81             break;
82
83         case '7':
84             CL;
85             if (data_.size() == 0) {

```

```

86         std::cout << "Nothing to delete. No multisets available." << std::endl
           ;
87         W;
88         break;
89     }
90     deleteMultiset();
91     W;
92     break;
93
94     case '8':
95         CL;
96         reset();
97         std::cout << "Resetted." << std::endl;
98         W;
99         break;
100
101     case '0':
102         return 0;
103
104     default:
105         std::cout << "Invalid option. Please, try again!\n";
106         goto Again;
107     }
108 }
109 }

```

%enditemize

4 Результаты работы программы

Разработанная программа имеет консольный интерфейс и предоставляет пользователю меню, позволяющее поэтапно выполнять все действия.

Меню программы

После запуска программы отображается главное меню, через которое пользователь может:

- создать новое мультимножество (ручное или случайное заполнение);
- вывести список всех созданных мультимножеств;
- выбрать два мультимножества для выполнения операций;
- выполнить все операции сразу или выбрать отдельную операцию;
- выйти из программы.

Создание мультимножеств

Программа поддерживает два режима создания мультимножеств:

- **ручной режим:** пользователь вводит коды Грея и их кратности, программа проверяет корректность кода и не позволяет превысить максимальную кратность;
- **автоматический режим:** программа случайным образом заполняет мультимножество элементами универсума, не превышая заданные ограничения; пользователю необходимо ввести имя и мощность.

Также для быстрого создания мультимножеств можно использовать создание случайного мультимножества: в таком случае у него будут случайные имя и содержимое, что особенно удобно для быстрой генерации.

Работа с несколькими мультимножествами

Программа может одновременно хранить много мультимножеств, каждому из которых присваивается уникальное имя. В любой момент пользователь может вывести названия и содержимое всех мультимножеств. Это особенно удобно, если пользователь забыл их названия: с помощью этого списка он сможет найти нужные ему мультимножества и провести над ними необходимые операции.

Обработка некорректного ввода

Программа устойчиво ведёт себя при некорректных данных:

- При вводе неверных команд в меню отображается предупреждение и предлагается повторить ввод;
- при попытке ввести код, отсутствующий в универсуме, выводится сообщение об ошибке, а также список возможных кодов (сам универсум);
- при попытке превысить максимальную кратность программа блокирует добавление;
- при попытке обращения к несуществующему мультимножеству отображается предупреждение и предлагается повторить ввод;
- при попытке создать мультимножество с уже существующим именем программа предлагает повторить ввод и т.д.

Работа с пустыми множествами

Программа корректно обрабатывает пустые мультимножества: все операции (объединение, пересечение, разность и др.) с пустыми мультимножествами дают ожидаемый математический результат, что демонстрирует устойчивость реализации.

Пример работы

При запуске программы высвечивается приветственное окно. Чтобы продолжить в главное меню и начать пользоваться основным функционалом программы, надо нажать на любую кнопку (рис. 1).

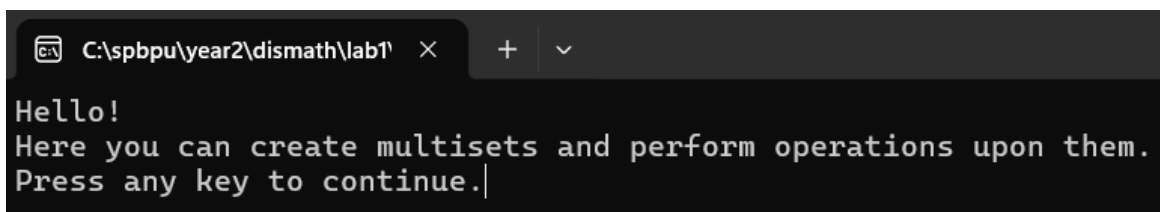
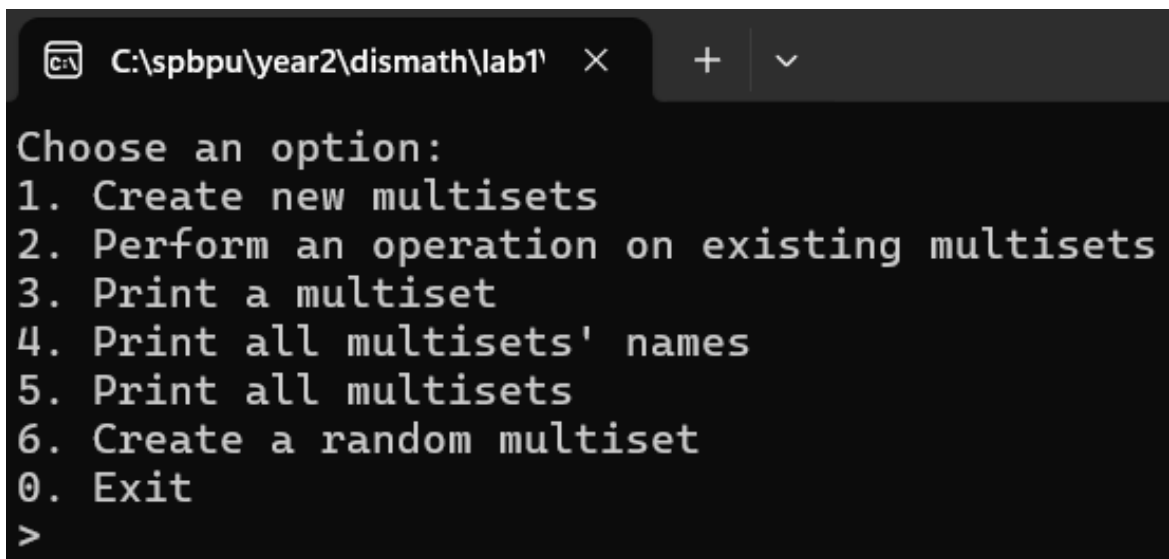


Рис. 1: Приветственное сообщение при запуске программы.

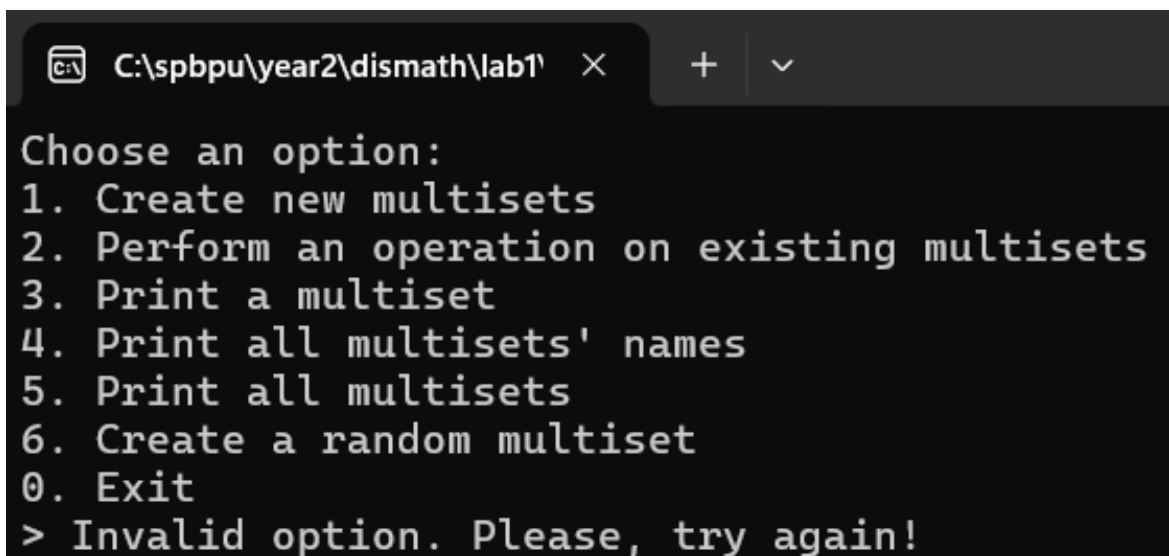
После этого пользователь попадает в главное меню, где ему предлагается ряд опций на выбор. Чтобы выбрать нужную опцию, пользователю нужно ввести соответствующую цифру (рис. 2).



```
C:\spbpu\year2\dismath\lab1' X + v
Choose an option:
1. Create new multisets
2. Perform an operation on existing multisets
3. Print a multiset
4. Print all multisets' names
5. Print all multisets
6. Create a random multiset
0. Exit
>
```

Рис. 2: Меню программы.

Если пользователь введет некорректный вариант, программа об этом сообщит и предложит выбрать снова (рис. 3).



```
C:\spbpu\year2\dismath\lab1' X + v
Choose an option:
1. Create new multisets
2. Perform an operation on existing multisets
3. Print a multiset
4. Print all multisets' names
5. Print all multisets
6. Create a random multiset
0. Exit
> Invalid option. Please, try again!
```

Рис. 3: Обработка некорректного пользовательского ввода.

Если пользователь решит выполнить какие-то операции над мульти-множествами, пока они не заданы, программа сообщит об этом и попросит сначала создать мультимножества (рис. 4).

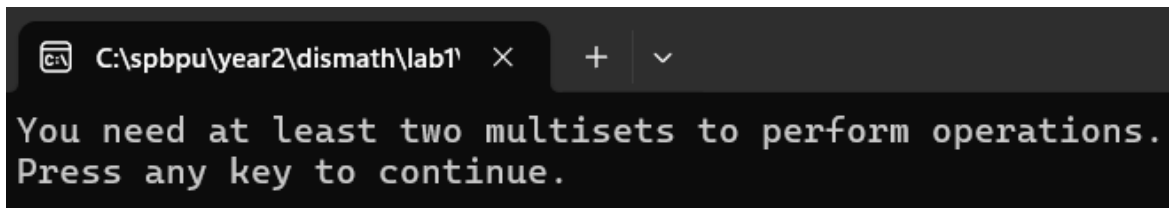
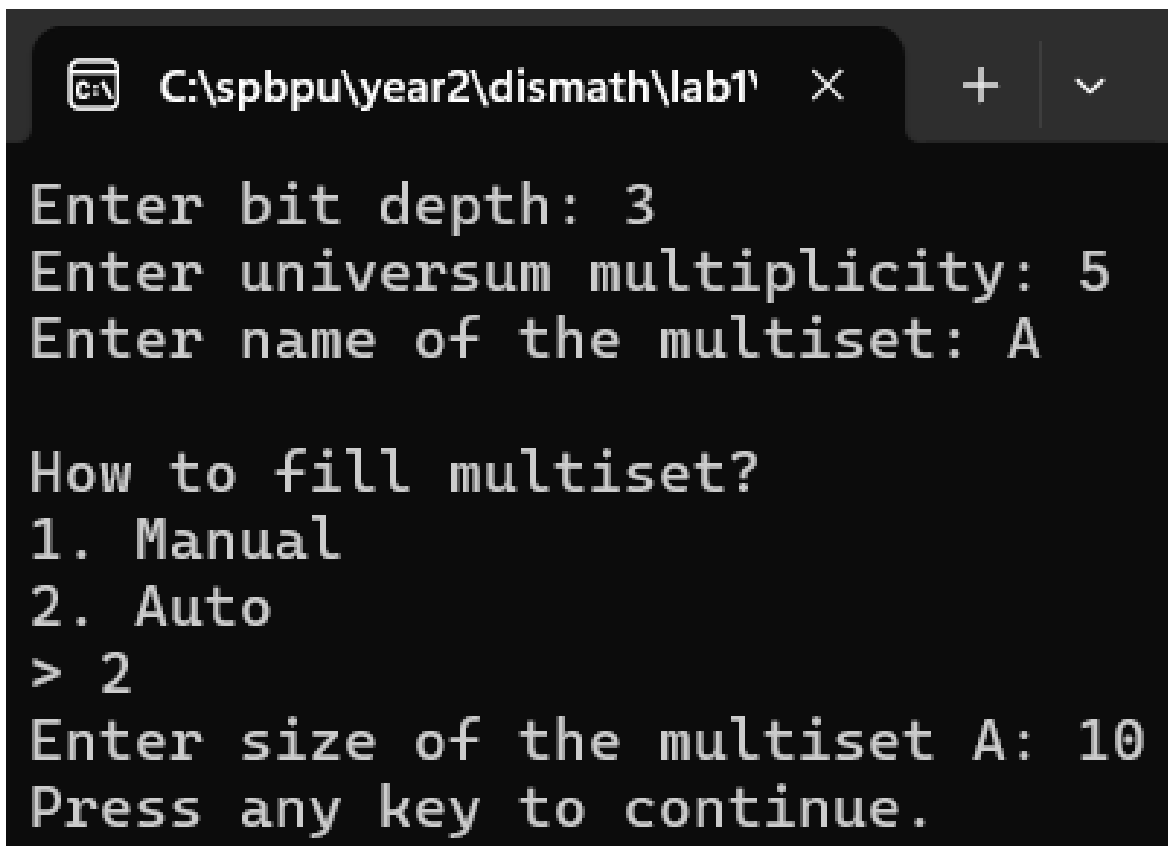


Рис. 4: Обработка попытки проведения операций над несуществующими мультимножествами.

Процесс создания мультимножеств включает в себя несколько этапов (рис. 5).

1. Ввод разрядности, если универсум еще не создан (например, 3).
2. Ввод кратности элементов универсума, если он еще не создан (например, 5).
3. Название мультимножества (например, A).
4. Выбор заполнения мультимножества: ручной или автоматический.
5. Ввод мощности мультимножества (например, 10).

A screenshot of a Windows command prompt window. The title bar shows the file path 'C:\spbp\year2\dismath\lab1' and standard window controls. The command prompt displays the following text:

```
Enter bit depth: 3
Enter universum multiplicity: 5
Enter name of the multiset: A

How to fill multiset?
1. Manual
2. Auto
> 2
Enter size of the multiset A: 10
Press any key to continue.
```

Рис. 5: Создание мультимножества с автоматическим заполнением.

При ручном наборе элементов мультимножества пользователь может вводить как указать кратность текущего элемента, так и опустить ее (в таком случае будет использовано значение кратности по умолчанию - 1). Если такой элемент уже существует, кратности будут сложены. Если в процессе создания мультимножества будут введены неверные данные (например, слишком большая кратность), программа сообщит об этом и попросит ввести корректные данные (рис. 6).

```
C:\spbp\year2\dismath\lab1' X + v
Enter name of the multiset: B
How to fill multiset?
1. Manual
2. Auto
> 1
Enter size of the multiset B: 7
Fill Multiset B:
Enter 7 Gray codes (limit: 5)
(You can enter either Gray code only or Gray code [space] count)
Code #1: 10000000000000
Invalid code. Valid: 000 001 011 010 110 111 101 100
Try again.
Code #1: 100
Code #2: 101 2
Code #4: 100 3
Code #7: 011
Press any key to continue.
```

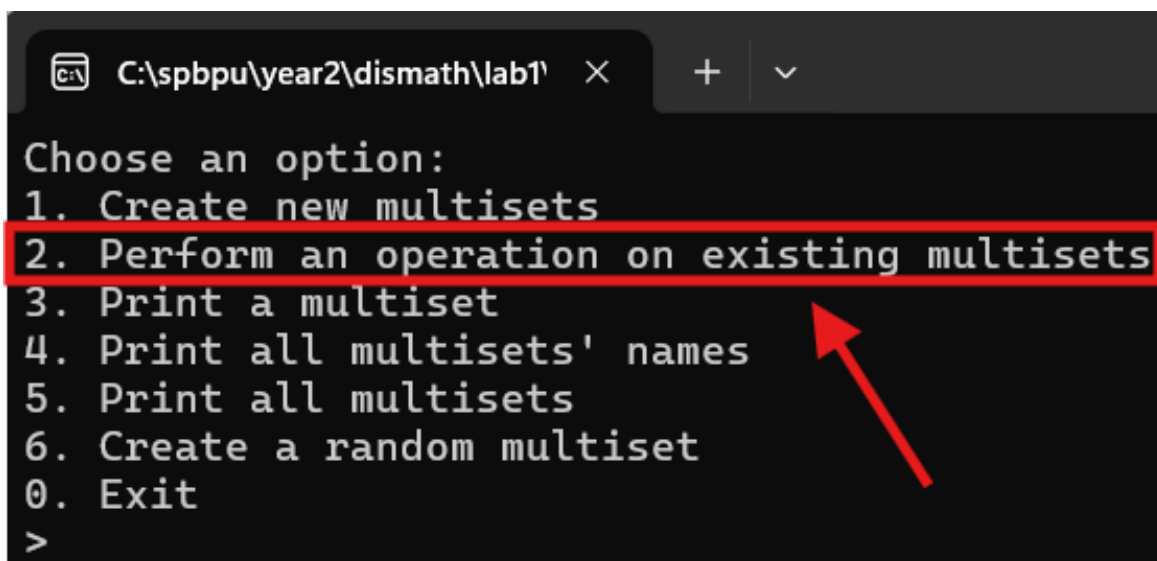
Рис. 6: Создание мультимножества с ручным вводом элементов.

После того как пользователь создаст хотя бы одно мультимножество, он сможет использовать весь функционал программы, в том числе операции над ними, поскольку вместе с первым мультимножеством создается и универсум, который можно увидеть, например, при выборе опции 5 "Print all multisets"(рис. 7).

```
C:\spbp\year2\dismath\lab1' X + v
#1 "A": { "000"^1, "001"^2, "010"^1, "011"^1, "101"^3, "110"^2}
#2 "B": { "011"^1, "100"^4, "101"^2}
#3 "U": { "000"^5, "001"^5, "010"^5, "011"^5, "100"^5, "101"^5, "110"^5, "111"^5}
Press any key to continue...
```

Рис. 7: Вывод всех мультимножеств.

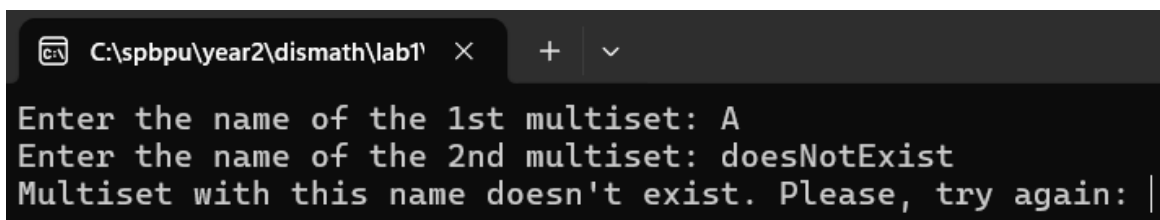
Для проведения операций над мультимножествами пользователю надо выбрать пункт 2 в меню (рис. 8).



```
C:\spbpu\year2\dismath\lab1' X + v
Choose an option:
1. Create new multisets
2. Perform an operation on existing multisets
3. Print a multiset
4. Print all multisets' names
5. Print all multisets
6. Create a random multiset
0. Exit
>
```

Рис. 8: Опция для проведения операций над мультимножествами.

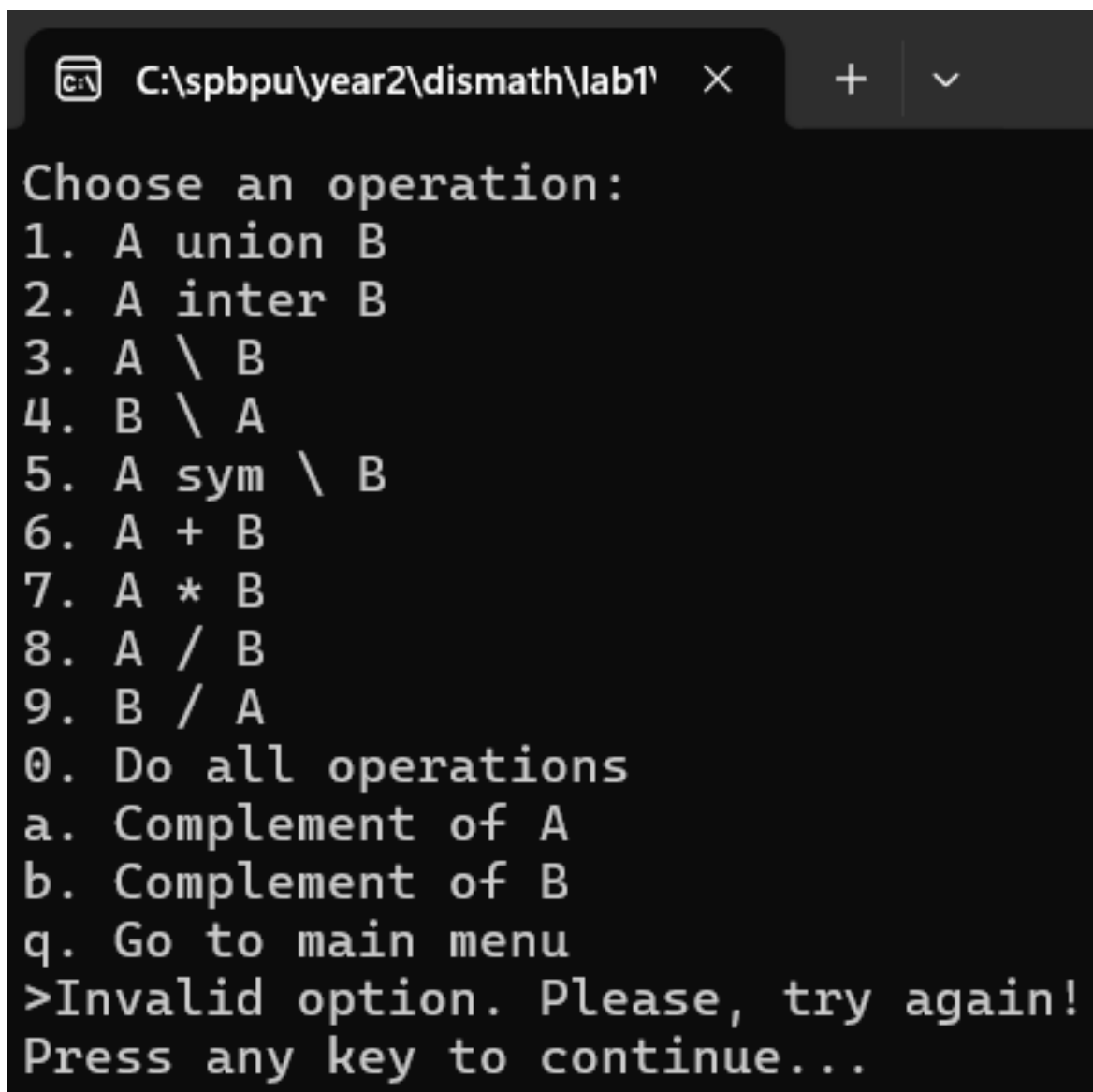
Далее ему будет предложено ввести названия мультимножеств, с которыми будут проводиться операции. При неверном названии будет предупреждение от программы и просьба ввести корректное название (рис. 9).



```
C:\spbpu\year2\dismath\lab1' X + v
Enter the name of the 1st multiset: A
Enter the name of the 2nd multiset: doesNotExist
Multiset with this name doesn't exist. Please, try again: |
```

Рис. 9: Ввод названий мультимножеств для проведения операций.

После ввода названий мультимножеств пользователю предоставляется список возможных операций над ними. Чтобы выбрать операцию, пользователю нужно нажать на соответствующую клавишу. При неверном выборе программа попросит пользователя выбрать корректный вариант (рис. 10).



```
C:\spbpu\year2\dismath\lab1' X + v

Choose an operation:
1. A union B
2. A inter B
3. A \ B
4. B \ A
5. A sym \ B
6. A + B
7. A * B
8. A / B
9. B / A
0. Do all operations
a. Complement of A
b. Complement of B
q. Go to main menu
>Invalid option. Please, try again!
Press any key to continue...
```

Рис. 10: Неверный выбор операции над мультимножествами.

Для удобства пользователю предоставляется вариант "Do all operations"("Сделать все операции"). При выборе этого варианта пользователь получит результаты всех предложенных операций над этими мультимножествами (рис. 11).

```
C:\spbp\year2\dismath\lab1 × + v
===== MULTISETS =====
U: { "000"^10, "001"^10, "011"^10, "010"^10, "110"^10, "111"^10, "101"^10, "100"^10}
A: { "000"^1, "001"^2, "010"^1, "011"^1, "101"^3, "110"^2}
B: { "011"^1, "100"^4, "101"^2}

===== OPERATIONS =====
A union B: { "000"^1, "001"^2, "010"^1, "011"^1, "100"^4, "101"^3, "110"^2}
A inter B: { "011"^1, "101"^2}
A \ B: { "000"^1, "001"^2, "010"^1, "101"^1, "110"^2}
B \ A: { "100"^4}
A sym \ B: { "000"^1, "001"^2, "010"^1, "100"^4, "101"^1, "110"^2}
A + B: { "000"^1, "001"^2, "010"^1, "011"^2, "100"^4, "101"^5, "110"^2}
A * B: { "011"^1, "101"^5}
A / B: { "011"^1, "101"^1}
B / A: { "011"^1, "101"^0}
Complement of A: { "000"^4, "001"^3, "010"^4, "011"^4, "100"^5, "101"^2, "110"^3, "111"^5}
Complement of B: { "000"^5, "001"^5, "010"^5, "011"^4, "100"^1, "101"^3, "110"^5, "111"^5}
Press any key to continue...
```

Рис. 11: Выполнение всех операций над мультимножествами.

Результаты подтвердили, что все операции над мультимножествами реализованы корректно.

Заключение

В ходе лабораторной работы была разработана программа, позволяющая:

- генерировать бинарный код Грея заданной разрядности;
- формировать мультимножества двумя способами (ручной и автоматический);
- выполнять над мультимножествами различные операции:
 - объединение,
 - пересечение,
 - разность,
 - симметрическая разность,
 - дополнение,
 - арифметическая сумма,
 - арифметическая разность,
 - арифметическое произведение,
 - арифметическое деление;
- обеспечивать защиту от некорректного пользовательского ввода.

Программа демонстрирует корректную реализацию всех заданных функций. Программа создана по принципам ООП: выделены отдельные классы `Multiset` и `Interface`, что удобно разделяет интерфейс и функционал каждого из класса и позволяет вносить изменения в программу легче, чем если бы она имела процедурную структуру. Консольный интерфейс позволяет легко

создавать, удалять мультимножества и проводить над ними операции; помимо этого, интерфейс распознает некорректный ввод пользователя и корректно его обрабатывает.

Плюсы программы.

- Использование объектно-ориентированной парадигмы программирования, что позволяет удобно модифицировать программу.
- Возможность работы над несколькими мультимножествами без стирания предыдущих из памяти.
- Возможность удаления мультимножеств.

Минусы программы.

- Все данные хранятся в оперативной памяти, без сохранения на диск.
- Ограничения по размеру входных данных (из-за возможного целочисленного переполнения при больших объемах).
- Использование контейнеров STL, что ухудшает производительность и повышает расходы памяти.

Масштабирование. Используемая архитектура позволяет легко добавлять новые операции над мультимножествами (например, вычисление мощности или сортировку по частоте). Также возможна интеграция с графическим интерфейсом или веб-интерфейсом. В будущем можно добавить возможность работы с тернарными операциями.

В перспективе программа может быть:

- использована как часть библиотеки по дискретной математике;

- интегрирована в учебные курсы для демонстрации операций над мультимножествами;
- адаптирована под параллельные вычисления для одновременных операций над большими мультимножествами или для ускорения самих операций.

Таким образом, реализованная программа является масштабируемой и обладает потенциалом развития. Её структура позволяет расширять функциональность и использовать в более крупных программных системах.

Список литературы

- [1] Павловская Т. А., Щюпак Ю. А. С++ Объектно-ориентированное программирование: Практикум. — СПб.: Питер, 2006. — 265 с.
- [2] Коды Грэя и задачи перебора (Электронный ресурс).
URL: <https://habr.com/ru/articles/200806/>
(дата обращения: 13.09.2025).
- [3] Секция "Телематика" (Электронный ресурс).
URL: <https://tema.spbstu.ru/dismath/>
(дата обращения: 13.09.2025).
- [4] Класс map | Microsoft Learn (Электронный ресурс).
URL: <https://learn.microsoft.com/ru-ru/cpp/standard-library/map-class?view=msvc-170>
(дата обращения: 18.09.2025).

Приложение 1. Файл Multiset.h.

```
1 #include <string>
2 #include <map>
3 #include <vector>
4
5 class Multiset {
6 private:
7     std::map<std::string, int> elements; // Multiset
8     static std::vector<std::string> universe; // universum
9     static int maxMultiplicity; // universum's multiplicity
10
11 public:
12     // setting up universum and the multisets
13     static std::vector<std::string> generateGrayCode(int n);
14     static void createUniverse(int depth);
15     static void setMaxMultiplicity(int limit);
16     static int getMaxMultiplicity();
17     int size() const;
18     void fillManual(int size);
19     void fillRandom(int size);
20
21     // Modifying multisets methods
22     int frequency(const std::string& code) const;
23     bool canAdd(const std::string& code) const;
24     bool isValidCode(const std::string& code) const;
25     int add(const std::string& code, int count);
26
27     // Operations with multisets
28     Multiset unionWith(const Multiset& other) const;
29     Multiset intersectionWith(const Multiset& other) const;
30     Multiset differenceWith(const Multiset& other) const;
```

```
31     Multiset arithmeticDifferenceWith(const Multiset& other) const;
32     Multiset symmetricDifferenceWith(const Multiset& other) const;
33     Multiset arithmeticSum(const Multiset& other) const;
34     Multiset arithmeticProduct(const Multiset& other) const;
35     Multiset arithmeticDivision(const Multiset& other) const;
36     Multiset complement() const;
37
38     // Printing
39     void print() const;
40     void printU() const;
41     static Multiset getUniverse();
42 };
```

ЛИСТИНГ 1: Файл Multiset.h

Приложение 2. Файл Multiset.cpp.

```
1 #include "Multiset.h"
2 #include <iostream>
3 #include <stdexcept>
4 #include <sstream>
5
6 int Multiset::maxMultiplicity;
7
8 void Multiset::setMaxMultiplicity(int limit) {
9     maxMultiplicity = limit;
10 }
11
12 int Multiset::getMaxMultiplicity() {
13     return maxMultiplicity;
14 }
15
16 std::vector<std::string> Multiset::generateGrayCode(int n) {
17     if (n <= 0) return {};
18     std::vector<std::string> gray;
19     int total = 1 << n;
20
21     for (int i = 0; i < total; ++i) {
22         int grayCode = i ^ (i >> 1);
23         std::string code = "";
24         for (int j = n - 1; j >= 0; --j) {
25             code += (grayCode & (1 << j)) ? '1' : '0';
26         }
27         gray.push_back(code);
28     }
29     return gray;
30 }
```

```

31
32 std::vector<std::string> Multiset::universe; // default; will change in Interface
33
34 void Multiset::createUniverse(int depth) {
35     universe = generateGrayCode(depth);
36 }
37
38 bool Multiset::isValidCode(const std::string& code) const {
39     for (const auto& u : universe) {
40         if (code == u) return true;
41     }
42     return false;
43 }
44
45 bool Multiset::canAdd(const std::string& code) const {
46     // Check if code is valid and if adding one more would exceed maxMultiplicity
47     if (!isValidCode(code)) return false;
48     int current = frequency(code);
49     return current < maxMultiplicity;
50 }
51
52 int Multiset::frequency(const std::string& code) const {
53     auto it = elements.find(code);
54     return (it != elements.end()) ? it->second : 0;
55 }
56
57 int Multiset::add(const std::string& code, int count = 1) {
58     /*
59     * Adds up to 'count' instances of 'code', but not exceeding maxMultiplicity.
60     * Returns the actual number of instances added (which may be less than 'count' or ==
61     * zero).
62     */

```

```

62     if (!isValidCode(code) || count <= 0) return 0;
63
64     int current = frequency(code);
65     int available = maxMultiplicity - current;
66     int actualToAdd = std::min(count, available);
67
68     if (actualToAdd > 0) {
69         elements[code] += actualToAdd;
70     }
71
72     return actualToAdd;
73 }
74
75 void Multiset::fillManual(int size) {
76     std::cin.ignore();
77     if (size < 0) return;
78
79     if (size != 0)
80         std::cout << "Enter " << size << " Gray codes (limit: " << maxMultiplicity << ")
81             << "\n"
82             << "(You can enter either Gray code only or Gray code [space] count; other
83             << "numbers won't be accounted)" << std::endl;
84
85     for (int i = 0; i < size; ) {
86         std::string code;
87         while (true) {
88             std::cout << "Code #" << (i + 1) << ": ";
89
90             std::string line;
91             std::getline(std::cin, line);
92             std::istringstream iss(line);

```



```

92     std::string code;
93     if (!(iss >> code)) {
94         std::cout << "Empty input. Try again.\n";
95         continue;
96     }
97
98     int count = 1;
99     iss >> count;
100
101     if (!isValidCode(code)) {
102         std::cout << "Invalid code. Valid: ";
103         for (const auto& c : universe) std::cout << c << " ";
104         std::cout << "\nTry again.\n";
105         continue;
106     }
107
108     if (!canAdd(code)) {
109         std::cout << "You've reached your limit(" << maxMultiplicity
110             << ") for the code \"" << code << "\". Please pick different.\n";
111         continue;
112     }
113
114     int currentCount = frequency(code);
115     if (currentCount + count > maxMultiplicity) {
116         std::cout << "Adding " << count << " of \"" << code
117             << "\" would exceed limit (" << maxMultiplicity
118             << "). Currently have " << currentCount << ".\n"
119             << "Try again with fewer or different code.\n";
120         continue;
121     }
122
123     for (int j = 0; j < count; ++j) {

```

```

124         add(code);
125     }
126
127     i += count;
128     break;
129 }
130 }
131 }
132
133 void Multiset::fillRandom(int size) {
134     /*
135      * Fills the multiset with 'size' random codes from the universe,
136      * and ensures not to exceed maxMultiplicity for any code.
137      */
138
139     if (size <= 0 || universe.empty()) return;
140
141     for (int i = 0; i < size; ++i) {
142         std::string code;
143         do {
144             code = universe[rand() % universe.size()];
145         } while (!canAdd(code));
146
147         add(code);
148     }
149 }
150
151 int Multiset::size() const {
152     int total = 0;
153     for (const auto& p : elements) total += p.second;
154     return total;
155 }

```

```

156
157 Multiset Multiset::unionWith(const Multiset& other) const {
158     /*
159     * For union, we take the maximum frequency of each element from both multisets,
160     * and put it in the temporary multiset 'result'.
161     */
162     Multiset result;
163
164     for (const auto& p : elements) {
165         if (p.second > 0) {
166             result.elements[p.first] = p.second;
167         }
168     }
169     for (const auto& p : other.elements) {
170         if (p.second > 0) {
171             result.elements[p.first] = std::max(result.elements[p.first], p.second);
172         }
173     }
174     return result;
175 }
176
177 Multiset Multiset::intersectionWith(const Multiset& other) const {
178     Multiset result;
179
180     for (const auto& p : elements) {
181         if (other.elements.count(p.first)) {
182             result.elements[p.first] = std::min(p.second, other.elements.at(p.first));
183         }
184     }
185     return result;
186 }
187

```

```

188 Multiset Multiset::differenceWith(const Multiset& other) const {
189     return intersectionWith(other.complement());
190 }
191
192 Multiset Multiset::arithmeticDifferenceWith(const Multiset& other) const {
193     Multiset result;
194
195     for (const auto& p : elements) {
196         int count = p.second;
197         if (other.elements.count(p.first)) {
198             count -= other.elements.at(p.first);
199         }
200         if (count > 0) {
201             result.elements[p.first] = count;
202         }
203     }
204     return result;
205 }
206
207 Multiset Multiset::symmetricDifferenceWith(const Multiset& other) const {
208     Multiset result;
209     auto uni = unionWith(other);
210     auto inter = intersectionWith(other);
211     result = uni.differenceWith(inter);
212
213     return result;
214 }
215
216 Multiset Multiset::arithmeticSum(const Multiset& other) const {
217     Multiset result;
218
219     for (const auto& p : elements) result.elements[p.first] += p.second;

```

```

220     for (const auto& p : other.elements) result.elements[p.first] += p.second;
221     for (auto& p : result.elements)
222         if (p.second > maxMultiplicity)
223             p.second = maxMultiplicity;
224
225     return result;
226 }
227
228 Multiset Multiset::arithmeticProduct(const Multiset& other) const {
229     Multiset result;
230
231     for (const auto& p : elements) {
232         if (other.elements.count(p.first)) {
233             result.elements[p.first] = std::min(p.second * other.elements.at(p.first),
234                                                 maxMultiplicity);
235         }
236     }
237
238     return result;
239 }
240
241 Multiset Multiset::arithmeticDivision(const Multiset& other) const {
242     Multiset result;
243
244     for (const auto& p : elements)
245         if (other.elements.count(p.first) && other.elements.at(p.first) > 0)
246             if (p.second / other.elements.at(p.first) > 0)
247                 result.elements[p.first] = p.second / other.elements.at(p.first);
248
249     return result;
250 }

```

```

251 Multiset Multiset::complement() const {
252     Multiset result;
253
254     for (const auto& code : universe) {
255         int current = frequency(code);
256         int needed = maxMultiplicity - current;
257         if (needed > 0) {
258             result.elements[code] = needed;
259         }
260     }
261
262     return result;
263 }
264
265 void Multiset::print() const {
266     std::cout << "{ ";
267     bool first = true;
268     for (const auto& p : elements) {
269         if (!first) std::cout << ", ";
270         std::cout << "\"" << p.first << "\"^" << p.second;
271         first = false;
272     }
273     std::cout << "}" << std::endl;
274 }
275
276 void Multiset::printU() const {
277     std::cout << "{ ";
278     bool first = true;
279     for (const auto& p : universe) {
280         if (!first) std::cout << ", ";
281         std::cout << "\"" << p << "\"^" << maxMultiplicity;
282         first = false;

```

```

283     }
284     std::cout << "}" << std::endl;
285 }
286
287 Multiset Multiset::getUniverse() {
288     Multiset uni;
289     for (const auto& elem : universe) {
290         uni.elements[elem] = maxMultiplicity;
291     }
292     return uni;
293 }

```

Листинг 2: Файл Multiset.cpp

Приложение 3. Файл Interface.h.

```
1 #pragma once
2 #include <string>
3 #include <map>
4
5 class Multiset;
6
7 class Interface
8 {
9 private:
10     std::map<std::string, Multiset> data_;
11     int uniMultiplicity_;
12     int bitDepth_;
13     int uniSize_;
14
15     bool isValidName(const std::string& name);
16     void createMultisets(bool rand = false);
17     void displayMenu();
18     void performAll(Multiset& A, Multiset& B, std::string& nameA, std::string& nameB);
19     void perform(std::string& nameA, std::string& nameB);
20     void printMultiset(std::string& name);
21     void printNames();
22     void printAllMultisets();
23     void createRandom();
24     void deleteMultiset();
25     void reset();
26
27 public:
28     int run();
29 };
```

Листинг 3: Файл Interface.h

Приложение 4. Файл Interface.cpp.

```
1  #include "Interface.h"
2  #include "Multiset.h"
3
4  #include <iostream>
5  #include <conio.h>
6  #include <string>
7
8  #define CL system("cls")
9  #define W std::cout << "Press any key to continue..."; trash = _getch()
10 #define IGN std::cin.clear(); std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n')
11
12 bool Interface::isValidName(const std::string& name) {
13     bool isValid = data_.find(name) == data_.end() && name != "";
14     if (isValid)
15         return true;
16     else
17         /*std::cout << "Multiset with this name already exists or name is invalid.\n"
18         << "Please, try again: ";*/
19     return false;
20 }
21
22 void Interface::createMultisets(bool rand) {
23     if (data_.size() == 0) { // Here we set the universe parameters
24         std::cout << "Enter bit depth: ";
25         int depth;
26         while (!(std::cin >> depth) || depth < 0) {
27             std::cout << "Invalid input. Enter non-negative integer: ";
28             IGN
29         }
```

```

30
31     bitDepth_ = depth;
32
33     if (bitDepth_ == 0)
34         goto AddUniverse;
35
36     Multiset::createUniverse(depth);
37
38     std::cout << "Enter universum multiplicity: ";
39     int multi;
40     IGN
41     while (!(std::cin >> multi) || multi < 0) {
42         std::cout << "Invalid input. Enter non-negative integer: ";
43         IGN
44     }
45     Multiset::setMaxMultiplicity(multi);
46     uniMultiplicity_ = multi;
47
48     AddUniverse:
49     if (uniMultiplicity_ == 0)
50         Multiset::createUniverse(0);
51     data_["U"] = Multiset::getUniverse();
52     uniSize_ = Multiset::getUniverse().size();
53 }
54
55 if (rand) {
56     createRandom();
57 }
58 else {
59     Multiset A;
60
61     std::cout << "Enter name of the multiset: ";

```

```

62     std::string name;
63     std::cin >> name;
64     while (!isValidName(name)) {
65         IGN
66         std::cin >> name;
67     }
68     if (bitDepth_ * uniMultiplicity_ == 0) {
69         goto AddMultiset;
70     }
71     std::cout << "\nHow to fill multiset?\n1. Manual\n2. Auto\n> ";
72     int choice;
73     IGN
74     while (!(std::cin >> choice) || (choice != 1 && choice != 2)) {
75         IGN
76         std::cout << "Enter 1 or 2: ";
77     }
78
79     int sizeA;
80     std::cout << "Enter size of the multiset " << name << ": ";
81     IGN
82     while (!(std::cin >> sizeA) || sizeA < 0 || sizeA > uniSize_) {
83         IGN
84         std::cout << "Size can't be negative or more than universe's size: ";
85     }
86
87     if (choice == 1) {
88         std::cout << "Fill Multiset " << name << ':' << std::endl;
89         A.fillManual(sizeA);
90     }
91     else {
92         A.fillRandom(sizeA);
93     }

```

```

94     AddMultiset:
95     data_[name] = A;
96 }
97 }
98
99 void Interface::performAll(Multiset& A, Multiset& B, std::string& nameA, std::string&
    nameB) {
100     std::cout << "\n===== MULTISETS =====\n";
101     std::cout << "U: "; A.printU(); std::cout << '\n';
102     std::cout << nameA << ": "; A.print(); std::cout << '\n';
103     std::cout << nameB << ": "; B.print(); std::cout << '\n';
104     std::cout << "\n===== OPERATIONS =====\n";
105     std::cout << nameA << " union " << nameB << ": "; A.unionWith(B).print(); std::
        cout << '\n';
106     std::cout << nameA << " inter " << nameB << ": "; A.intersectionWith(B).print(); std
        ::cout << '\n';
107     std::cout << nameA << " \ " << nameB << ": "; A.differenceWith(B).print(); std::
        cout << '\n';
108     std::cout << nameB << " \ " << nameA << ": "; B.differenceWith(A).print(); std::
        cout << '\n';
109     std::cout << nameA << " /\ " << nameB << ": "; A.symmetricDifferenceWith(B).
        print(); std::cout << '\n';
110     std::cout << nameA << " + " << nameB << ": "; A.arithmeticSum(B).print(); std::
        cout << '\n';
111     std::cout << nameA << " - " << nameB << ": "; A.arithmeticDifferenceWith(B).print
        (); std::cout << '\n';
112     std::cout << nameB << " - " << nameA << ": "; B.arithmeticDifferenceWith(A).print
        (); std::cout << '\n';
113     std::cout << nameA << " * " << nameB << ": "; A.arithmeticProduct(B).print(); std::
        cout << '\n';
114     std::cout << nameA << " / " << nameB << ": "; A.arithmeticDivision(B).print(); std::
        cout << '\n';

```

```

115     std::cout << nameB << " / " << nameA << ": "; B.arithmeticDivision(A).print(); std::
        cout << '\n';
116     std::cout << "Complement of " << nameA << ": "; auto compA = A.complement();
        compA.print(); std::cout << '\n';
117     std::cout << "Complement of " << nameB << ": "; auto compB = B.complement();
        compB.print(); std::cout << '\n';
118 }
119
120 void Interface::perform(std::string& nameA, std::string& nameB) {
121     std::cout << "Enter the name of the 1st multiset: ";
122     std::cin >> nameA;
123     while (isValidName(nameA)) {
124         std::cout << "Multiset with this name doesn't exist. Please, try again: ";
125         std::cin >> nameA;
126     }
127     std::cout << "Enter the name of the 2nd multiset: ";
128     std::cin >> nameB;
129     while (isValidName(nameB)) {
130         std::cout << "Multiset with this name doesn't exist. Please, try again: ";
131         std::cin >> nameB;
132     }
133
134     Multiset compA;
135     Multiset compB;
136     int trash;
137     int choice;
138
139     while (true) {
140         CL;
141         std::cout << "Choose an operation:\n"
142             << "1. ' " << nameA << "' union ' " << nameB << "'\n"
143             << "2. ' " << nameB << "' inter ' " << nameB << "'\n"

```

```

144         << "3. '" << nameA << "' \\' " << nameB << "'\n"
145         << "4. '" << nameB << "' \\' " << nameA << "'\n"
146         << "5. '" << nameA << "' /\\' " << nameB << "'\n"
147         << "6. '" << nameA << "' + '" << nameB << "'\n"
148         << "7. '" << nameA << "' - '" << nameB << "'\n"
149         << "8. '" << nameA << "' * '" << nameB << "'\n"
150         << "9. '" << nameA << "' / '" << nameB << "'\n"
151         << "0. '" << nameB << "' / '" << nameA << "'\n"
152         << "d. Do all operations\n"
153         << "a. Complement of '" << nameA << "'\n"
154         << "b. Complement of '" << nameB << "'\n"
155         << "q. Go to main menu\n>";
156
157     choice = _getch();
158
159     switch (choice) {
160     case '1':
161         std::cout << "A union B: "; data_[nameA].unionWith(data_[nameB]).print();
162         std::cout << '\n';
163         W;
164         break;
165     case '2':
166         std::cout << "A inter B: "; data_[nameA].intersectionWith(data_[nameB]).print
167         (); std::cout << '\n';
168         W;
169         break;
170     case '3':
171         std::cout << "A \\' B: "; data_[nameA].arithmeticDifferenceWith(data_[nameB
172         ]).print(); std::cout << '\n';
173         W;
174         break;
175     case '4':

```

```

173         std::cout << "B \\\ A: "; data_[nameB].arithmeticDifferenceWith(data_[nameA
           ]).print(); std::cout << '\n';
174     W;
175     break;
176 case '5':
177     std::cout << "A sym \\\ B: "; data_[nameA].symmetricDifferenceWith(data_[
           nameB]).print(); std::cout << '\n';
178     W;
179     break;
180 case '6':
181     std::cout << "A + B: "; data_[nameA].arithmeticSum(data_[nameB]).print();
           std::cout << '\n';
182     W;
183     break;
184 case '7':
185     std::cout << "A - B: "; data_[nameA].differenceWith(data_[nameB]).print(); std
           ::cout << '\n';
186     W;
187     break;
188 case '8':
189     std::cout << "A * B: "; data_[nameA].arithmeticProduct(data_[nameB]).print()
           ; std::cout << '\n';
190     W;
191     break;
192 case '9':
193     std::cout << "A / B: "; data_[nameA].arithmeticDivision(data_[nameB]).print()
           ; std::cout << '\n';
194     W;
195     break;
196 case '0':
197     std::cout << "B / A: "; data_[nameB].arithmeticDivision(data_[nameA]).print()
           ; std::cout << '\n';

```

```

198         W;
199         break;
200     case 'd':
201         performAll(data_[nameA], data_[nameB], nameA, nameB);
202         W;
203         break;
204     case 'a':
205         std::cout << "Complement of A: "; compA = data_[nameA].complement();
206         compA.print(); std::cout << '\n';
207         W;
208         break;
209     case 'b':
210         std::cout << "Complement of B: "; compB = data_[nameB].complement();
211         compB.print(); std::cout << '\n';
212         W;
213         break;
214     case 'q':
215         goto Break;
216     default:
217         std::cout << "Invalid option. Please, try again!\n";
218         W;
219     }
220     continue;
221 Break:
222     break;
223 }
224 void Interface::printMultiset(std::string& name) {
225     data_[name].print();
226 }
227

```



```

228 void Interface::printNames() {
229     int i = 1;
230     for (const auto& pair : data_) {
231         std::cout << "#" << i << ": " << pair.first << std::endl;
232         i++;
233     }
234 }
235
236 void Interface::printAllMultisets() {
237     int i = 1;
238     for (const auto& pair : data_) {
239         std::cout << '#' << i << " \'" << pair.first << "\': ";
240         pair.second.print();
241         i++;
242     }
243 }
244
245 void Interface::createRandom() {
246     std::string name;
247     int i = 10000;
248     do { name = std::to_string(rand() % i++); } while (!isValidName(name));
249     Multiset A;
250     i = 100;
251     int size = rand() % i;
252     while (size > uniMultiplicity_ * bitDepth_)
253         size--;
254     A.fillRandom(size);
255     data_[name] = A;
256     std::cout << "Random multiset created.\n"
257         << "Name: " << name
258         << "\nSize: " << A.size() << std::endl;
259 }

```

```

260
261 void Interface::displayMenu() {
262     std::cout << "Choose an option:\n"
263         << "1. Create new multisets\n"
264         << "2. Perform an operation on existing multisets\n"
265         << "3. Print a multiset\n"
266         << "4. Print all multisets' names\n"
267         << "5. Print all multisets\n"
268         << "6. Create a random multiset\n"
269         << "7. Delete a multiset\n"
270         << "8. Reset (delete all multisets and universe)\n"
271         << "0. Exit\n> ";
272 }
273
274 void Interface::deleteMultiset() {
275     std::string name;
276     std::cout << "Enter the name of the multiset to delete: ";
277     std::cin >> name;
278     while (isValidName(name) || name == "U") { // Checks if it DOESN'T exist yet.
279         if (isValidName(name))
280             std::cout << "Multiset with this name doesn't exist. Please, try again: ";
281         else
282             std::cout << "Can't delete universe. Please, try again: ";
283         std::cin >> name;
284     }
285     data_.erase(name);
286     std::cout << "Deleted." << std::endl;
287 }
288
289 void Interface::reset() {
290     data_.clear();
291     bitDepth_ = 0;

```

```

292     uniMultiplicity_ = 0;
293 }
294
295 int Interface::run() {
296     int trash;
297     std::string nameA;
298     std::string nameB;
299
300     std::cout << "Hello!\n"
301         << "Here you can create multisets and perform operations upon them.\n"
302         << "Press any key to continue.";
303     trash = _getch();
304
305     while (true) {
306         CL;
307         displayMenu();
308         Again:
309         char option = _getch();
310         switch (option) {
311             case '1':
312                 CL;
313                 createMultisets();
314                 std::cout << "Press any key to continue.";
315                 trash = _getch();
316                 break;
317
318             case '2':
319                 CL;
320                 if (data_.size() < 2) {
321                     std::cout << "You need at least two multisets to perform operations.\n"
322                         << "Press any key to continue.";
323                     trash = _getch();

```

```

324         break;
325     }
326     perform(nameA, nameB);
327     break;
328
329 case '3':
330     CL;
331     if (data_.size() == 0) {
332         std::cout << "Nothing to print. No multisets available." << std::endl;
333         W;
334         break;
335     }
336     std::cout << "Enter the name of a multiset to print: ";
337     std::cin >> nameA;
338     while (isValidName(nameA)) { // Checks if it DOESN'T exist yet.
339         std::cout << "Wrong name. Please try again: ";
340         std::cin >> nameA;
341     }
342     printMultiset(nameA);
343     W;
344     break;
345
346 case '4':
347     CL;
348     if (data_.size() == 0) {
349         std::cout << "Nothing to print. No multisets available." << std::endl;
350         W;
351         break;
352     }
353     printNames();
354     W;
355     break;

```

```

356
357     case '5':
358         CL;
359         if (data_.size() == 0) {
360             std::cout << "Nothing to print. No multisets available." << std::endl;
361             W;
362             break;
363         }
364         printAllMultisets();
365         W;
366         break;
367
368     case '6':
369         CL;
370         if (data_.size() == 0)
371             createMultisets(true);
372         else
373             createRandom();
374         W;
375         break;
376
377     case '7':
378         CL;
379         if (data_.size() == 0) {
380             std::cout << "Nothing to delete. No multisets available." << std::endl;
381             W;
382             break;
383         }
384         deleteMultiset();
385         W;
386         break;
387

```

```
388     case '8':
389         CL;
390         reset();
391         std::cout << "Resetted." << std::endl;
392         W;
393         break;
394
395     case '0':
396         return 0;
397
398     default:
399         std::cout << "Invalid option. Please, try again!\n";
400         goto Again;
401     }
402 }
403 }
```

Листинг 4: Файл Interface.cpp

Приложение 5. Файл main.cpp.

```
1 #include <iostream>
2 #include "Multiset.h"
3 #include "Interface.h"
4
5 int main() {
6     Interface menu;
7     menu.run();
8
9     return 0;
10 }
```

Листинг 5: Файл main.cpp