

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 группы НОМЕР ГРУППЫ

ФИО

Принял:

ФИО

«_____» _____ 2025 г.

Санкт-Петербург, 2025

Содержание

Введение	4
1 Постановка задачи	5
2 Математическое описание	6
3 Особенности реализации	8
3.1 Класс Multiset	8
3.2 Класс Interface	22
3.3 Файл main.cpp	36
4 Результаты работы программы	37
4.1 Меню программы	37
4.2 Создание мультимножеств	37
4.3 Работа с несколькими мультимножествами	38
4.4 Обработка некорректного ввода	38
4.5 Работа с пустыми множествами	39
4.6 Пример работы	39
Заключение	47
Список литературы	49
Приложение	51
Приложение 1. Файл Multiset.h	51
Приложение 1. Файл Multiset.cpp	53
Приложение 3. Файл Interface.h	63

Приложение 4. Файл Interface.cpp	64
Приложение 5. Файл main.cpp	76

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:

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

- Пересечение:

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

- Разность:

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

- **Симметрическая разность:**

$$A \triangle B = \{x \mid (x \in A \wedge x \notin B) \vee (x \notin A \wedge x \in B)\}.$$

- **Дополнение:**

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

- **Арифметическая сумма:**

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

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

- **Арифметическая разность:**

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

- **Арифметическое произведение:**

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

- **Арифметическое деление:**

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

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

3.1 Класс Multiset

- Метод `generateGrayCode`

Вход: `int n`.

Выход: `std::vector<std::string>`.

Назначение: Генерирует коды Грея разрядности n .

Код:

```
1      std::vector<std::string> Multiset::generateGrayCode(int n) {
2          if (n <= 0) return {};
3          std::vector<std::string> gray;
4          int total = 1 << n;
5
6          for (int i = 0; i < total; ++i) {
7              int grayCode = i ^ (i >> 1);
8              std::string code = "";
9              for (int j = n - 1; j >= 0; --j) {
10                 code += (grayCode & (1 << j)) ? '1' : '0';
11             }
12             gray.push_back(code);
13         }
14         return gray;
15     }
```

- Метод `createUniverse`

Вход: `int depth`.

Выход: отсутствует.

Назначение: Создаёт универсум по кодам Грея.

Код:

```

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

```

- **Метод** `setMaxMultiplicity`

Вход: `int limit`.

Выход: отсутствует.

Назначение: Устанавливает максимальную кратность.

Код:

```

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

```

- **Метод** `getMaxMultiplicity`

Вход: отсутствует.

Выход: `int`.

Назначение: Возвращает текущую максимальную кратность.

Код:

```

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

```

- **Метод** `size`

Вход: отсутствует.

Выход: `int`.

Назначение: Возвращает суммарную 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`.

Выход: отсутствует.

Назначение: Заполняет мультимножество вручную.

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: " <<
7                 maxMultiplicity << ")\n"
8                 << "(You can enter either Gray code only or Gray code [space]
9                 count; other numbers won't be accounted)" << std::endl;
10
11         for (int i = 0; i < size; ) {
12             std::string code;
13             while (true) {
14                 std::cout << "Code #" << (i + 1) << ": ";
15
16                 std::string line;
17                 std::getline(std::cin, line);
18                 std::istringstream iss(line);
19
20                 std::string code;
21                 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(" <<
36             maxMultiplicity
37             << ") for the code \"" << code << "\". Please pick
38             different.\n";
39         continue;
40     }
41
42     int currentCount = frequency(code);
43     if (currentCount + count > maxMultiplicity) {
44         std::cout << "Adding " << count << " of \"" <<
45             code
46             << "\" would exceed limit (" << maxMultiplicity
47             << "). Currently have " << currentCount << ".\n"
48             << "Try again with fewer or different code.\n";
49         continue;
50     }

```

```

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`.

Выход: отсутствует.

Назначение: Заполняет случайными элементами.

Код:

```

1  void Multiset::fillRandom(int size) {
2      /*
3       * Fills the multiset with 'size' random codes from the universe,
4       * and ensures not to exceed maxMultiplicity for any code.
5       */
6
7      if (size <= 0 || universe.empty()) return;
8
9      for (int i = 0; i < size; ++i) {
10         std::string code;
11         do {
12             code = universe[rand() % universe.size()];
13         } while (!canAdd(code));
14
15         add(code);
16     }

```

17 }

- Метод `frequency`

Вход: `const std::string& code`.

Выход: `int`.

Назначение: Возвращает кратность элемента.

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`.

Выход: `bool`.

Назначение: Проверяет, можно ли добавить элемент.

Код:

```
1         bool Multiset::canAdd(const std::string& code) const {
2             // Check if code is valid and if adding one more would exceed
3             // maxMultiplicity
4             if (!isValidCode(code)) return false;
5             int current = frequency(code);
6             return current < maxMultiplicity;
7         }
```

- Метод `isValidCode`

Вход: `const std::string& code`.

Выход: `bool`.

Назначение: Проверяет, 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 умолчанию. 1).

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

Назначение: Добавляет элементы в массив.

Код:

```
1         int Multiset::add(const std::string& code, int count = 1) {
2             /*
3              * Adds up to 'count' instances of 'code', but not exceeding
4              * maxMultiplicity.
5              * Returns the actual number of instances added (which may be
6              * less than 'count' or == zero).
7              */
8             if (!isValidCode(code) || count <= 0) return 0;
9
10            int current = frequency(code);
11            int available = maxMultiplicity - current;
12            int actualToAdd = std::min(count, available);
13
14            if (actualToAdd > 0) {
15                elements[code] += actualToAdd;
16            }
```

```

15
16         return actualToAdd;
17     }

```

- Метод `unionWith`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Объединение (максимум кратностей).

Код:

```

1      Multiset Multiset::unionWith(const Multiset& other) const {
2          /*
3           * For union, we take the maximum frequency of each element from
4           * both multisets,
5           * and put it in the temporary multiset 'result'.
6           */
7          Multiset result;
8          result.universe = universe;
9
10         for (const auto& p : elements) {
11             if (p.second > 0) {
12                 result.elements[p.first] = p.second;
13             }
14         }
15         for (const auto& p : other.elements) {
16             if (p.second > 0) {
17                 result.elements[p.first] = std::max(result.elements[p.first],
18                                                         p.second);
19             }
20         }
21         return result;
22     }

```

- Метод `intersectionWith`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Пересечение (минимум кратностей).

Код:

```
1      Multiset Multiset::intersectionWith(const Multiset& other) const {
2          Multiset result;
3          result.universe = universe;
4
5          for (const auto& p : elements) {
6              if (other.elements.count(p.first)) {
7                  result.elements[p.first] = std::min(p.second, other.elements
8                      .at(p.first));
9              }
10         }
11         return result;
12     }
```

- Метод `differenceWith`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Разность $A \setminus B$.

Код:

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

- Метод `symmetricDifferenceWith`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Симметрическая разность.

Код:

```
1      Multiset Multiset::symmetricDifferenceWith(const Multiset& other)
2      {
3      auto diffA = arithmeticDifferenceWith(other);
4      auto diffB = other.arithmeticDifferenceWith(*this);
5
6      Multiset result;
7      result.universe = universe;
8
9      for (const auto& p : diffA.elements) result.elements[p.first] = p.
10         second;
11     for (const auto& p : diffB.elements) result.elements[p.first] = p.
12         second;
13
14     return result;
15 }
```

- **Метод** `arithmeticSum`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Арифметическая сумма.

Код:

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

```

6         for (const auto& p : other.elements) result.elements[p.first] += p.
           second;
7
8         return result;
9     }

```

- **Метод** `arithmeticDifferenceWith`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: арифметическая разность.

Код:

```

1         Multiset Multiset::arithmeticDifferenceWith(const Multiset& other)
           const {
2             Multiset result;
3             result.universe = universe;
4
5             for (const auto& p : elements) {
6                 int count = p.second;
7                 if (other.elements.count(p.first)) {
8                     count -= other.elements.at(p.first);
9                 }
10                if (count > 0) {
11                    result.elements[p.first] = count;
12                }
13            }
14            return result;
15        }

```

- **Метод** `arithmeticProduct`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Арифметическое произведение.

Код:

```
1      Multiset Multiset::arithmeticProduct(const Multiset& other) const {
2          Multiset result;
3          result.universe = universe;
4
5          for (const auto& p : elements) {
6              if (other.elements.count(p.first)) {
7                  result.elements[p.first] = std::min(p.second * other.
                        elements.at(p.first), maxMultiplicity);
8              }
9          }
10
11         return result;
12     }
```

- **Метод** `arithmeticDivision`

Вход: другое мультимножество.

Выход: новое мультимножество.

Назначение: Арифметическое деление.

Код:

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

```

10         return result;
11     }

```

- Метод `complement`

Вход: отсутствует.

Выход: новое мультимножество.

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        }

```

- Методы `print` и `printU`

Вход: отсутствует.

Выход: отсутствует.

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    }
11
12    void Multiset::printU() const {
13        std::cout << "{ ";
14        bool first = true;
15        for (const auto& p : universe) {
16            if (!first) std::cout << ", ";
17            std::cout << "\"" << p << "\"^" << 10;
18            first = false;
19        }
20        std::cout << "}" << std::endl;
21    }

```

- Метод `getUniverse`

Вход: отсутствует.

Выход: мультимножество.

Назначение: Возвращает универсум.

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`.

Выход: `bool`.

Назначение: Проверяет, валидно ли имя мультимножества.

Код:

```
1         bool Interface::isValidName(const std::string& name) {  
2             return data_.find(name) == data_.end() && name != "";  
3         }
```

- Метод canAdd

Вход: `const std::string& name`.

Выход: `bool`.

Назначение: Проверяет возможность использования имени.

Код:

```
1         bool Interface::canAdd(const std::string& name) {  
2             if (!isValidName(name)) {  
3                 std::cout << "Multiset with this name already exists or the  
4                     name is invalid.\n";  
5                 return false;  
6             }  
7             return true;  
            }
```

- Метод createMultisets

Вход: `bool rand`.

Выход: отсутствует.

Назначение: Создаёт новое мультимножество (ручное или случайное).

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                  std::cin.clear();
8                  std::cin.ignore(std::numeric_limits<std::streamsize>::max
9                      (), '\n');
10             }
11
12             bitDepth_ = depth;
13
14             if (bitDepth_ == 0)
15                 goto AddUniverse;
16
17             Multiset::createUniverse(depth);
18
19             std::cout << "Enter universum multiplicity: ";
20             int multi;
21             while (!(std::cin >> multi) || multi < 0) {
22                 std::cout << "Invalid input. Enter non-negative integer: ";
23                 std::cin.clear();
24                 std::cin.ignore(std::numeric_limits<std::streamsize>::max
25                     (), '\n');
26             }
27             Multiset::setMaxMultiplicity(multi);
28             uniMultiplicity_ = multi;
29
30             AddUniverse:
31             if (uniMultiplicity_ == 0)
```

```

30         Multiset::createUniverse(0);
31         data_["U"] = Multiset::getUniverse();
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 (!canAdd(name)) {
44             std::cout << "The name is already taken or invalid.
45                 Please, try again: ";
46             std::cin >> name;
47         }
48         if (bitDepth_ * uniMultiplicity_ == 0) {
49             goto AddMultiset;
50         }
51         std::cout << "\nHow to fill multiset?\n1. Manual\n2. Auto\n
52             > ";
53         int choice;
54         while (!(std::cin >> choice) || (choice != 1 && choice != 2)) {
55             std::cout << "Enter 1 or 2: ";
56         }
57
58         int sizeA;
59         std::cout << "Enter size of the multiset " << name << ": ";
60         while (!(std::cin >> sizeA) || sizeA < 0 || sizeA > bitDepth_
61             * uniMultiplicity_) {

```



```

59         std::cout << "Size can't be negative or more than
           universe's size: ";
60     }
61
62     if (choice == 1) {
63         std::cout << "Fill Multiset " << name << ':' << std::
           endl;
64         A.fillManual(sizeA);
65     }
66     else {
67         A.fillRandom(sizeA);
68     }
69     AddMultiset:
70     data_[name] = A;
71 }
72 }

```

- **Метод** `displayMenu`

Вход: отсутствует.

Выход: отсутствует.

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         << "0. Exit\n> ";

```

10

}

- Метод `performAll`

Вход: два мультимножества.

Выход: отсутствует.

Назначение: Выполняет все операции и выводит результаты.

Код:

```

1      void Interface::performAll(Multiset& A, Multiset& B) {
2          std::cout << "\n===== MULTISETS =====\n";
3          std::cout << "U: "; A.printU(); std::cout << '\n';
4          std::cout << "A: "; A.print(); std::cout << '\n';
5          std::cout << "B: "; B.print(); std::cout << '\n';
6
7          std::cout << "\n===== OPERATIONS =====\n";
8          std::cout << "A union B: "; A.unionWith(B).print(); std::cout
9              << '\n';
10         std::cout << "A inter B: "; A.intersectionWith(B).print(); std::
11             cout << '\n';
12         std::cout << "A \ B: "; A.differenceWith(B).print(); std::cout
13             << '\n';
14         std::cout << "B \ A: "; B.differenceWith(A).print(); std::cout
15             << '\n';
16         std::cout << "A sym \ B: "; A.symmetricDifferenceWith(B).
17             print(); std::cout << '\n';
18         std::cout << "A + B: "; A.arithmeticSum(B).print(); std::cout
19             << '\n';
20         std::cout << "A - B: "; A.arithmeticDifferenceWith(B).print();
21             std::cout << '\n';
22         std::cout << "B - A: "; B.arithmeticDifferenceWith(A).print();
23             std::cout << '\n';
24         std::cout << "A * B: "; A.arithmeticProduct(B).print(); std::cout

```

```

17         << '\n';
        std::cout << "A / B: "; A.arithmeticDivision(B).print(); std::cout
        << '\n';
18        std::cout << "B / A: "; B.arithmeticDivision(A).print(); std::cout
        << '\n';
19        std::cout << "Complement of A: "; auto compA = A.complement
        (); compA.print(); std::cout << '\n';
20        std::cout << "Complement of B: "; auto compB = B.complement
        (); compB.print(); std::cout << '\n';
21    }

```

- Метод `perform`

Вход: имена множеств.

Выход: отсутствует.

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,
6                try again: ";
7            std::cin >> nameA;
8        }
9        std::cout << "Enter the name of the 2nd multiset: ";
10       std::cin >> nameB;
11       while (isValidName(nameB)) {
12           std::cout << "Multiset with this name doesn't exist. Please,
13               try again: ";
14           std::cin >> nameB;
15       }

```

```

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. A union B\n"
24             << "2. A inter B\n"
25             << "3. A \ \ B\n"
26             << "4. B \ \ A\n"
27             << "5. A sym \ \ B\n"
28             << "6. A + B\n"
29             << "7. A - B\n"
30             << "8. A * B\n"
31             << "9. A / B\n"
32             << "0. B / A\n"
33             << "d. Do all operations\n"
34             << "a. Complement of A\n"
35             << "b. Complement of B\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(
43                         data_[nameB]).print(); std::cout << '\n';
44                     W;
45                     break;

```

```

45         case '2':
46             std::cout << "A inter B: "; data_[nameA].
                intersectionWith(data_[nameB]).print(); std::cout <<
                    '\n';
47         W;
48         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]);
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 }
103 }

```

- Метод `printMultiset`

Вход: имя множества.

Выход: отсутствует.

Назначение: Выводит содержимое множества.

Код:

```

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

```

- Метод `printNames`

Вход: отсутствует.

Выход: отсутствует.

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: отсутствует.

Выход: отсутствует.

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: отсутствует.

Выход: отсутствует.

Назначение: Создаёт случайное множество.

Код:

```

1     void Interface::createRandom() {
2         std::string name;
3         int i = 10000;
4         do { name = std::to_string(rand() % i++); } while (!isValidName
5             (name));
6         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** `run`

Вход: отсутствует.

Выход: `int`.

Назначение: 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
8             them.\n"
9             << "Press any key to continue.";
10            trash = _getch();
11
12            while (true) {
13                CL;
14                displayMenu();
15                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
28                         perform operations.\n"
29                         << "Press any key to continue.";
30                     trash = _getch();
31                     break;
32                 }
33                 perform(nameA, nameB);
34                 break;
35
36             case '3':
37                 CL;
38                 if (data_.size() == 0) {
39                     std::cout << "Nothing to print. No multisets available
40                         ." << std::endl;
41                     W;
42                     break;
43                 }
44                 std::cout << "Enter the name of a multiset to print: ";
45                 std::cin >> nameA;
46                 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 '0':
84             return 0;
85
86         default:
87             std::cout << "Invalid option. Please, try again!\n";
88             goto Again;
89     }
90 }
91 }

```

3.3 Файл main.cpp

- **Функция** `main`

Вход: отсутствует.

Выход: `int`.

Назначение: Точка входа в программу. Создаёт меню и запускает работу.

Листинг:

```

1     int main() {
2         Interface menu;
3         menu.run();

```

```
4         return 0;
5     }
```

4 Результаты работы программы

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

4.1 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.

4.2 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.

4.3 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.

4.4 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.

4.5 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.

4.6 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).

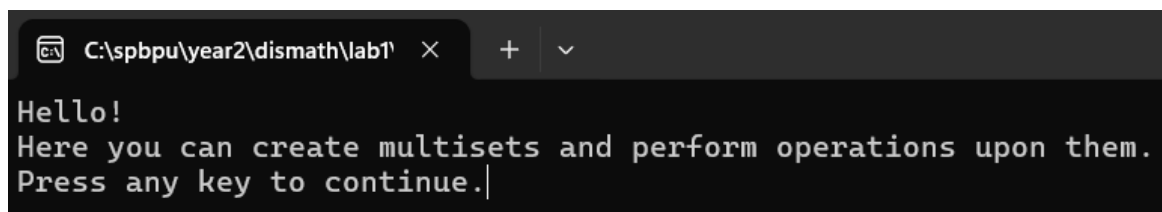
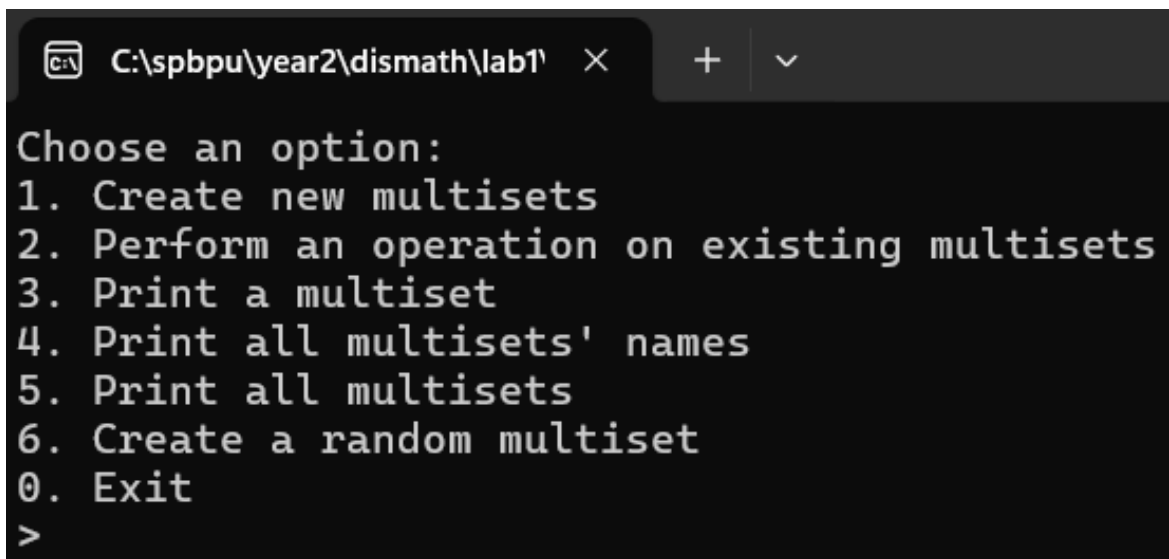


Рис. 1: Приветственное сообщение при запуске программы.

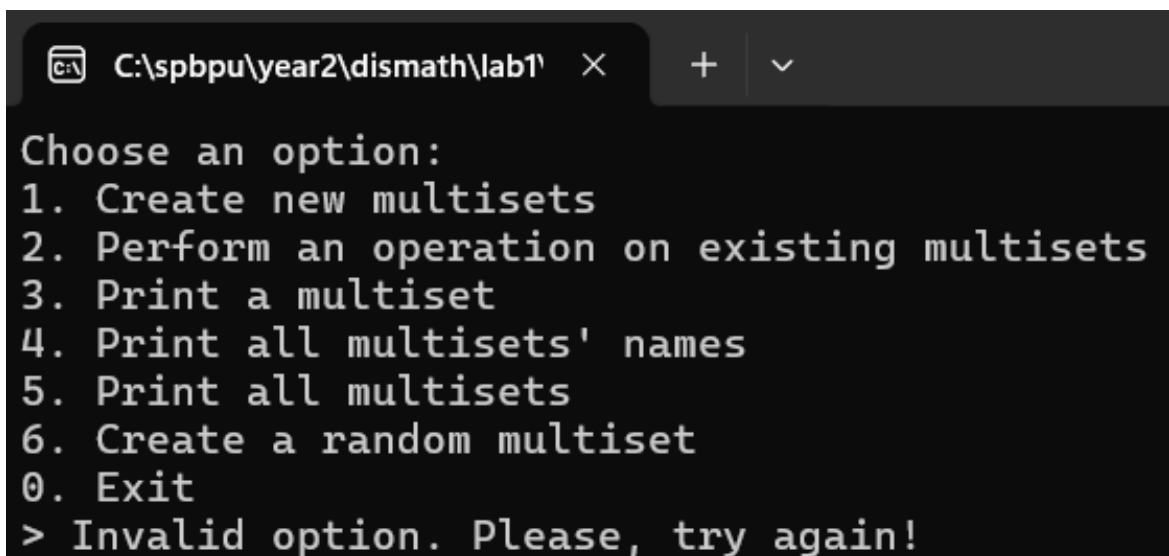
После этого пользователь попадает в главное меню, где ему предлагается ряд опций на выбор. Чтобы выбрать нужную опцию, пользователю нужно ввести соответствующую цифру (рис. 2).



```
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).

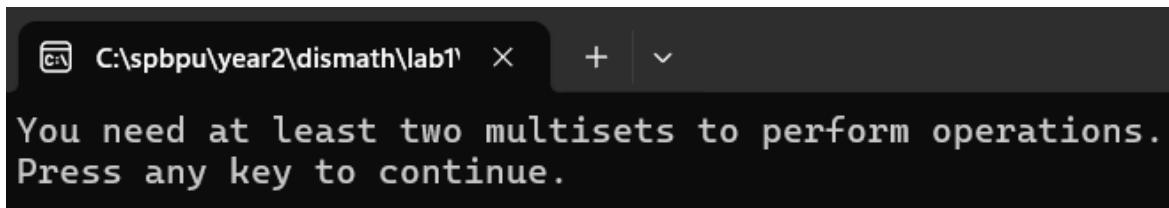
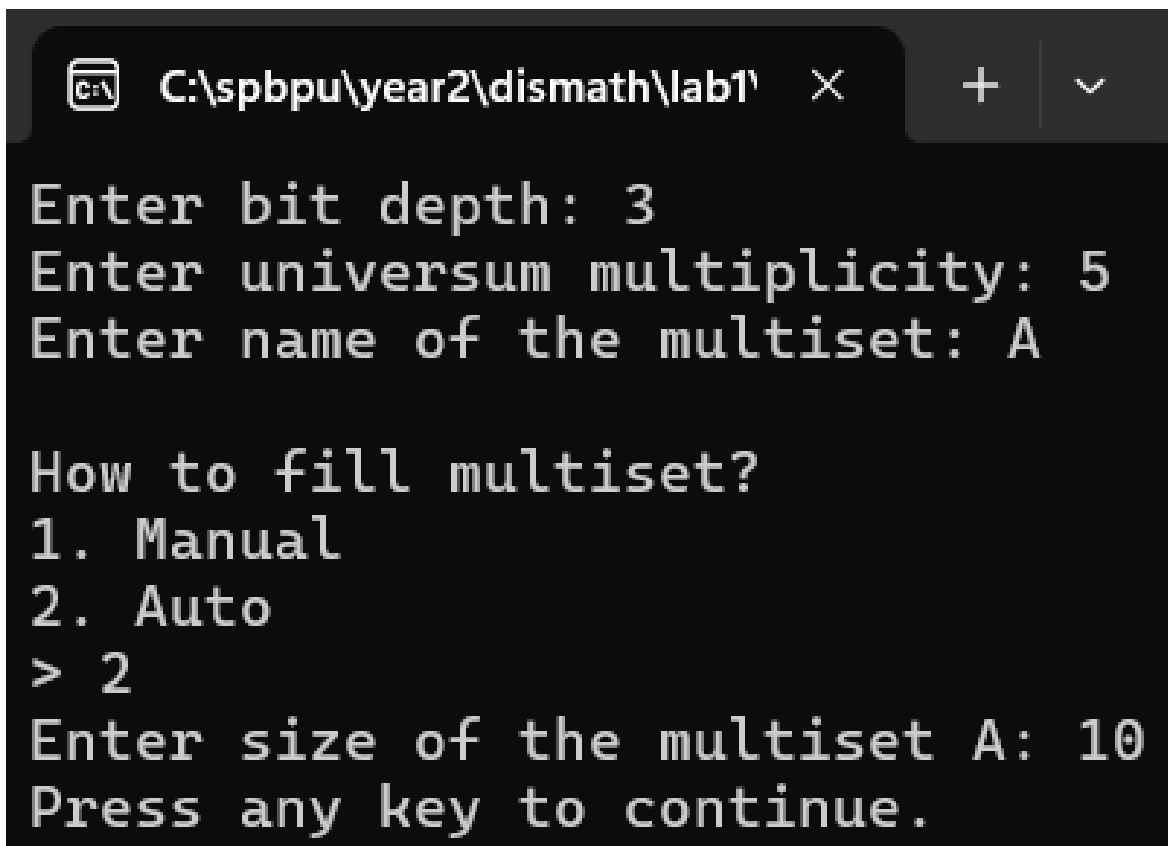


Рис. 4: Обработка попытки проведения операций над несуществующими мультимножествами.

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).



```
C:\spbpu\year2\dismath\lab1>
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' × + 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.
```

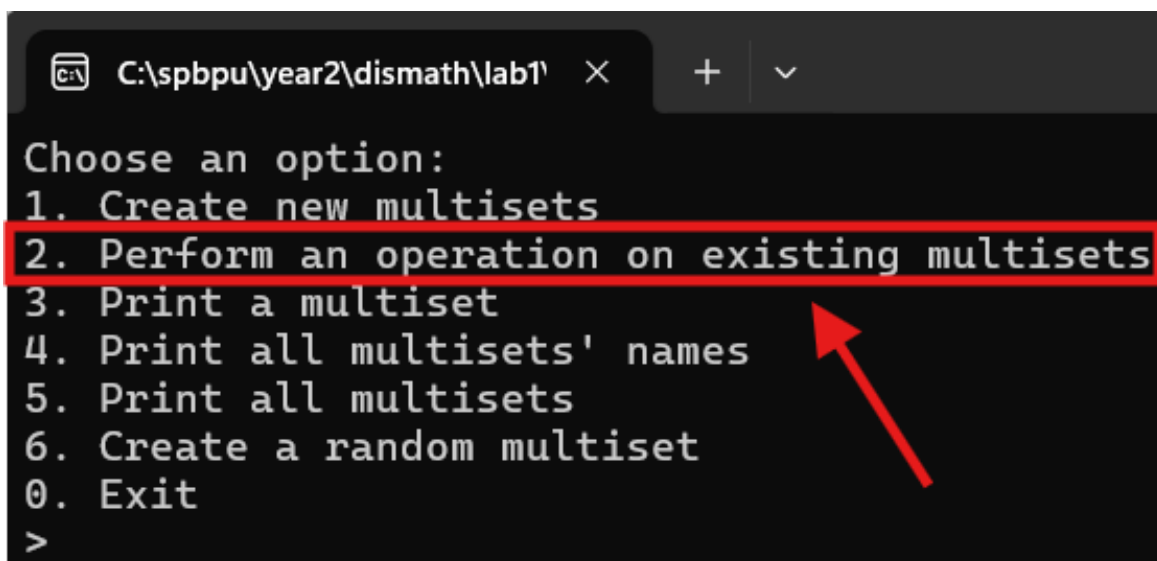
Рис. 6: Создание мультимножества с ручным вводом элементов.

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' × + 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.

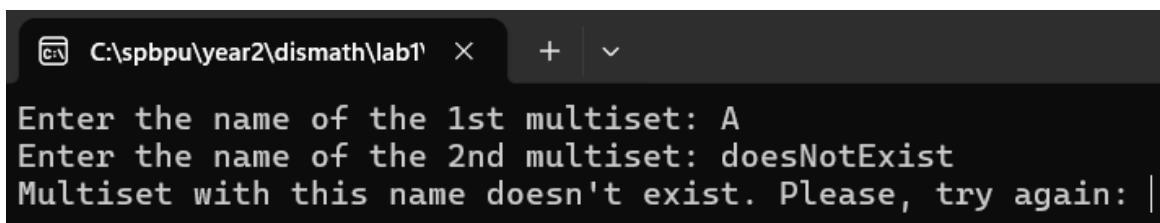
Для проведения операций над мультимножествами пользователю надо выбрать пункт 2 в меню (рис. 8).



```
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.

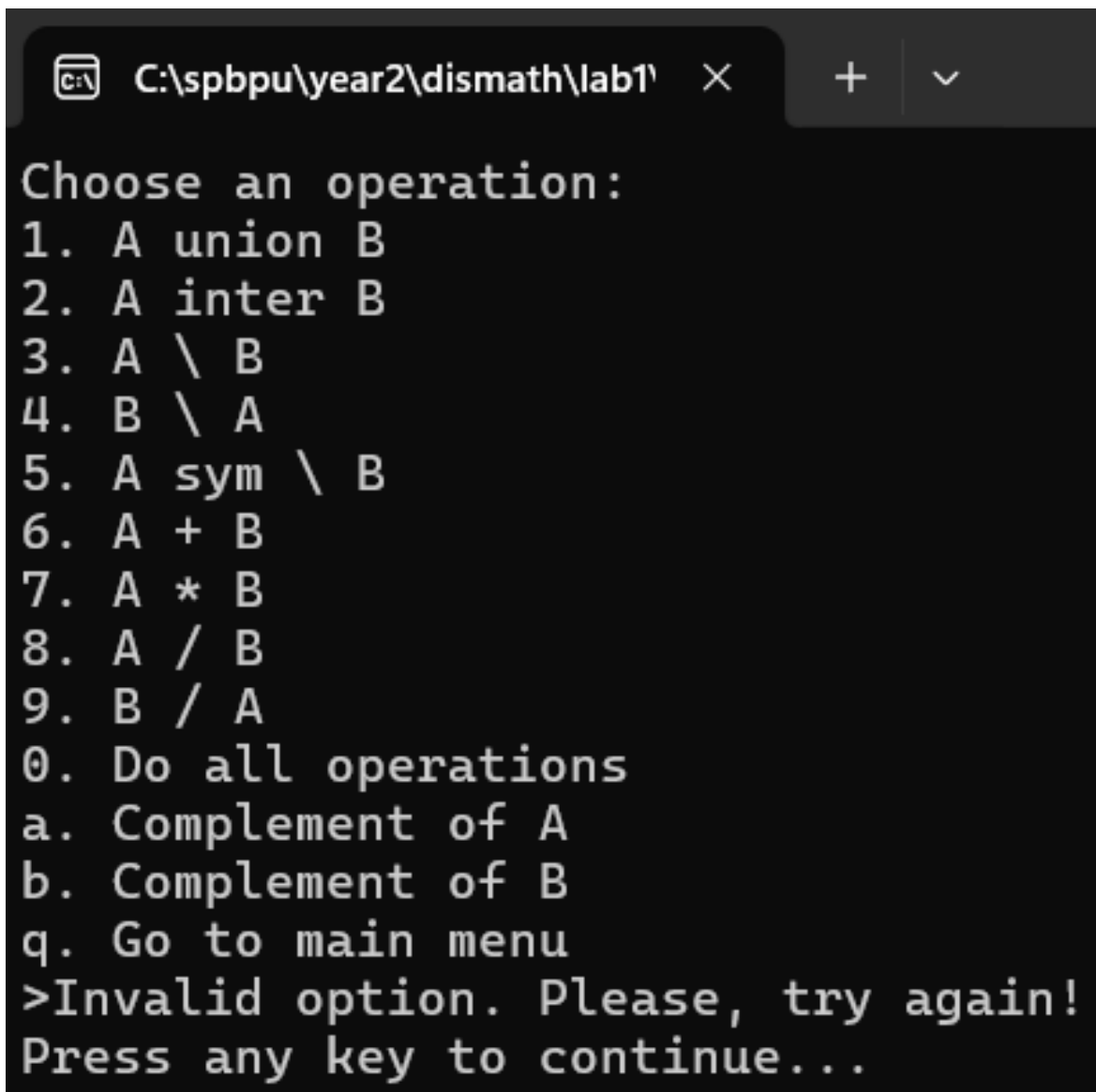
Далее ему будет предложено ввести названия мультимножеств, с которыми будут проводиться операции. При неверном названии будет предупреждение от программы и просьба ввести корректное название (рис. 9).



```
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;
- обеспечивать защиту от некорректного пользовательского ввода.

Оценка реализованной программы. Программа демонстрирует корректную реализацию всех заданных функций. Архитектура программы модульная: выделены отдельные классы `Multiset` и `Interface`, что способствует удобному сопровождению и расширению функционала. Интерфейс консольного меню позволяет легко управлять процессом, а наличие защиты от некорректного ввода делает программу устойчивой к ошибкам пользователя.

Плюсы программы.

- Использование объектно-ориентированной парадигмы программирования, что позволяет удобно модифицировать программу.
- Наличие всех основных операций над мультимножествами.
- Возможность ручного и автоматического заполнения.
- Возможность работы над несколькими мультимножествами без стирания предыдущих из памяти.

Минусы программы.

- Отсутствие графического интерфейса.
- Все данные хранятся в оперативной памяти, без сохранения на диск.
- Ограничения по размеру входных данных (из-за возможного целочисленного переполнения при больших объемах).

Перспективы развития. 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. Хранение данных в базе данных позволило бы работать с большими мультимножествами.

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

Литература

- [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

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 = 10;
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     */
63     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 << ")\n"
81             << "(You can enter either Gray code only or Gray code [space] count; other\n"
82             << "numbers won't be accounted)" << std::endl;
83
84     for (int i = 0; i < size; ) {
85         std::string code;
86         while (true) {
87             std::cout << "Code #" << (i + 1) << ": ";
88
89             std::string line;
90             std::getline(std::cin, line);
91             std::istringstream iss(line);
92
93             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     result.universe = universe;
164
165     for (const auto& p : elements) {
166         if (p.second > 0) {
167             result.elements[p.first] = p.second;
168         }
169     }
170     for (const auto& p : other.elements) {
171         if (p.second > 0) {
172             result.elements[p.first] = std::max(result.elements[p.first], p.second);
173         }
174     }
175     return result;
176 }
177
178 Multiset Multiset::intersectionWith(const Multiset& other) const {
179     Multiset result;
180     result.universe = universe;
181
182     for (const auto& p : elements) {
183         if (other.elements.count(p.first)) {
184             result.elements[p.first] = std::min(p.second, other.elements.at(p.first));
185         }
186     }
187     return result;
188 }

```

```

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

```

```

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

```

```

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

```

```

284     std::cout << "{ ";
285     bool first = true;
286     for (const auto& p : universe) {
287         if (!first) std::cout << ", ";
288         std::cout << "\" " << p << "\" ^" << 10;
289         first = false;
290     }
291     std::cout << "}" << std::endl;
292 }
293
294 Multiset Multiset::getUniverse() {
295     Multiset uni;
296     for (const auto& elem : universe) {
297         uni.elements[elem] = maxMultiplicity;
298     }
299     return uni;
300 }

```

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
14     bool isValidName(const std::string& name);
15     bool canAdd(const std::string& name);
16     void createMultisets(bool rand = false);
17     void displayMenu();
18     void performAll(Multiset& A, Multiset& B);
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
25 public:
26     int run();
27 };
```

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
11 bool Interface::isValidName(const std::string& name) {
12     return data_.find(name) == data_.end() && name != "";
13 }
14
15 bool Interface::canAdd(const std::string& name) {
16     if (!isValidName(name)) {
17         std::cout << "Multiset with this name already exists or the name is invalid.\n";
18         return false;
19     }
20     return true;
21 }
22
23 void Interface::createMultisets(bool rand) {
24     if (data_.size() == 0) { // Here we set the universe parameters
25         std::cout << "Enter bit depth: ";
26         int depth;
27         while (!(std::cin >> depth) || depth < 0) {
28             std::cout << "Invalid input. Enter non-negative integer: ";
29             std::cin.clear();
30             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
31         }
```



```

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

```

```

64     std::cin >> name;
65     while (!canAdd(name)) {
66         std::cout << "The name is already taken or invalid. Please, try again: ";
67         std::cin >> name;
68     }
69     if (bitDepth_ * uniMultiplicity_ == 0) {
70         goto AddMultiset;
71     }
72     std::cout << "\nHow to fill multiset?\n1. Manual\n2. Auto\n> ";
73     int choice;
74     while (!(std::cin >> choice) || (choice != 1 && choice != 2)) {
75         std::cout << "Enter 1 or 2: ";
76     }
77
78     int sizeA;
79     std::cout << "Enter size of the multiset " << name << ": ";
80     while (!(std::cin >> sizeA) || sizeA < 0 || sizeA > bitDepth_ * uniMultiplicity_) {
81         std::cout << "Size can't be negative or more than universe's size: ";
82     }
83
84     if (choice == 1) {
85         std::cout << "Fill Multiset " << name << ':' << std::endl;
86         A.fillManual(sizeA);
87     }
88     else {
89         A.fillRandom(sizeA);
90     }
91     AddMultiset:
92     data_[name] = A;
93 }
94 }
95

```

```

96 void Interface::performAll(Multiset& A, Multiset& B) {
97     std::cout << "\n===== MULTISSETS =====\n";
98     std::cout << "U: "; A.printU(); std::cout << '\n';
99     std::cout << "A: "; A.print(); std::cout << '\n';
100    std::cout << "B: "; B.print(); std::cout << '\n';
101
102    std::cout << "\n===== OPERATIONS =====\n";
103    std::cout << "A union B: "; A.unionWith(B).print(); std::cout << '\n';
104    std::cout << "A inter B: "; A.intersectionWith(B).print(); std::cout << '\n';
105    std::cout << "A \ B: "; A.differenceWith(B).print(); std::cout << '\n';
106    std::cout << "B \ A: "; B.differenceWith(A).print(); std::cout << '\n';
107    std::cout << "A sym \ B: "; A.symmetricDifferenceWith(B).print(); std::cout << '\n';
108    std::cout << "A + B: "; A.arithmeticSum(B).print(); std::cout << '\n';
109    std::cout << "A - B: "; A.arithmeticDifferenceWith(B).print(); std::cout << '\n';
110    std::cout << "B - A: "; B.arithmeticDifferenceWith(A).print(); std::cout << '\n';
111    std::cout << "A * B: "; A.arithmeticProduct(B).print(); std::cout << '\n';
112    std::cout << "A / B: "; A.arithmeticDivision(B).print(); std::cout << '\n';
113    std::cout << "B / A: "; B.arithmeticDivision(A).print(); std::cout << '\n';
114    std::cout << "Complement of A: "; auto compA = A.complement(); compA.print(); std::
        cout << '\n';
115    std::cout << "Complement of B: "; auto compB = B.complement(); compB.print(); std::
        cout << '\n';
116 }
117
118 void Interface::perform(std::string& nameA, std::string& nameB) {
119     std::cout << "Enter the name of the 1st multiset: ";
120     std::cin >> nameA;
121     while (isValidName(nameA)) {
122         std::cout << "Multiset with this name doesn't exist. Please, try again: ";
123         std::cin >> nameA;
124     }
125     std::cout << "Enter the name of the 2nd multiset: ";

```

```

126     std::cin >> nameB;
127     while (isValidName(nameB)) {
128         std::cout << "Multiset with this name doesn't exist. Please, try again: ";
129         std::cin >> nameB;
130     }
131
132     Multiset compA;
133     Multiset compB;
134     int trash;
135     int choice;
136
137     while (true) {
138         CL;
139         std::cout << "Choose an operation:\n"
140             << "1. A union B\n"
141             << "2. A inter B\n"
142             << "3. A \ \ B\n"
143             << "4. B \ \ A\n"
144             << "5. A sym \ \ B\n"
145             << "6. A + B\n"
146             << "7. A - B\n"
147             << "8. A * B\n"
148             << "9. A / B\n"
149             << "0. B / A\n"
150             << "d. Do all operations\n"
151             << "a. Complement of A\n"
152             << "b. Complement of B\n"
153             << "q. Go to main menu\n>";
154
155         choice = _getch();
156
157         switch (choice) {

```

```

158     case '1':
159         std::cout << "A union B: "; data_[nameA].unionWith(data_[nameB]).print();
160         std::cout << '\n';
161     W;
162     break;
163 case '2':
164     std::cout << "A inter B: "; data_[nameA].intersectionWith(data_[nameB]).print
165     (); std::cout << '\n';
166     W;
167     break;
168 case '3':
169     std::cout << "A \ B: "; data_[nameA].arithmeticDifferenceWith(data_[nameB
170     ]).print(); std::cout << '\n';
171     W;
172     break;
173 case '4':
174     std::cout << "B \ A: "; data_[nameB].arithmeticDifferenceWith(data_[nameA
175     ]).print(); std::cout << '\n';
176     W;
177     break;
178 case '5':
179     std::cout << "A sym \ B: "; data_[nameA].symmetricDifferenceWith(data_[
180     nameB]).print(); std::cout << '\n';
181     W;
182     break;
183 case '6':
184     std::cout << "A + B: "; data_[nameA].arithmeticSum(data_[nameB]).print();
185     std::cout << '\n';
186     W;
187     break;
188 case '7':

```

```

183         std::cout << "A - B: "; data_[nameA].differenceWith(data_[nameB]).print(); std
           ::cout << '\n';
184     W;
185     break;
186 case '8':
187     std::cout << "A * B: "; data_[nameA].arithmeticProduct(data_[nameB]).print()
           ; std::cout << '\n';
188     W;
189     break;
190 case '9':
191     std::cout << "A / B: "; data_[nameA].arithmeticDivision(data_[nameB]).print()
           ; std::cout << '\n';
192     W;
193     break;
194 case '0':
195     std::cout << "B / A: "; data_[nameB].arithmeticDivision(data_[nameA]).print()
           ; std::cout << '\n';
196     W;
197     break;
198 case 'd':
199     performAll(data_[nameA], data_[nameB]);
200     W;
201     break;
202 case 'a':
203     std::cout << "Complement of A: "; compA = data_[nameA].complement();
           compA.print(); std::cout << '\n';
204     W;
205     break;
206 case 'b':
207     std::cout << "Complement of B: "; compB = data_[nameB].complement();
           compB.print(); std::cout << '\n';
208     W;

```

```

209         break;
210     case 'q':
211         goto Break;
212     default:
213         std::cout << "Invalid option. Please, try again!\n";
214         W;
215     }
216     continue;
217 Break:
218     break;
219 }
220 }
221
222 void Interface::printMultiset(std::string& name) {
223     data_[name].print();
224 }
225
226 void Interface::printNames() {
227     int i = 1;
228     for (const auto& pair : data_) {
229         std::cout << "#" << i << ": " << pair.first << std::endl;
230         i++;
231     }
232 }
233
234 void Interface::printAllMultisets() {
235     int i = 1;
236     for (const auto& pair : data_) {
237         std::cout << '#' << i << " \n" << pair.first << "\n: ";
238         pair.second.print();
239         i++;
240     }

```

```

241 }
242
243 void Interface::createRandom() {
244     std::string name;
245     int i = 10000;
246     do { name = std::to_string(rand() % i++); } while (!isValidName(name));
247     Multiset A;
248     i = 100;
249     int size = rand() % i;
250     while (size > uniMultiplicity_ * bitDepth_)
251         size--;
252     A.fillRandom(size);
253     data_[name] = A;
254     std::cout << "Random multiset created.\n"
255         << "Name: " << name
256         << "\nSize: " << A.size() << std::endl;
257 }
258
259 void Interface::displayMenu() {
260     std::cout << "Choose an option:\n"
261         << "1. Create new multisets\n"
262         << "2. Perform an operation on existing multisets\n"
263         << "3. Print a multiset\n"
264         << "4. Print all multisets' names\n"
265         << "5. Print all multisets\n"
266         << "6. Create a random multiset\n"
267         << "0. Exit\n> ";
268 }
269
270 int Interface::run() {
271     int trash;
272     std::string nameA;

```



```

273     std::string nameB;
274
275     std::cout << "Hello!\n"
276         << "Here you can create multisets and perform operations upon them.\n"
277         << "Press any key to continue.";
278     trash = _getch();
279
280     while (true) {
281         CL;
282         displayMenu();
283         Again:
284         char option = _getch();
285         switch (option) {
286             case '1':
287                 CL;
288                 createMultisets();
289                 std::cout << "Press any key to continue.";
290                 trash = _getch();
291                 break;
292
293             case '2':
294                 CL;
295                 if (data_.size() < 2) {
296                     std::cout << "You need at least two multisets to perform operations.\n"
297                         << "Press any key to continue.";
298                     trash = _getch();
299                     break;
300                 }
301                 perform(nameA, nameB);
302                 break;
303
304             case '3':

```

```

305         CL;
306         if (data_.size() == 0) {
307             std::cout << "Nothing to print. No multisets available." << std::endl;
308             W;
309             break;
310         }
311         std::cout << "Enter the name of a multiset to print: ";
312         std::cin >> nameA;
313         while (isValidName(nameA)) { // Checks if it DOESN'T exist yet.
314             std::cout << "Wrong name. Please try again: ";
315             std::cin >> nameA;
316         }
317         printMultiset(nameA);
318         W;
319         break;
320
321     case '4':
322         CL;
323         if (data_.size() == 0) {
324             std::cout << "Nothing to print. No multisets available." << std::endl;
325             W;
326             break;
327         }
328         printNames();
329         W;
330         break;
331
332     case '5':
333         CL;
334         if (data_.size() == 0) {
335             std::cout << "Nothing to print. No multisets available." << std::endl;
336             W;

```

```

337         break;
338     }
339     printAllMultisets();
340     W;
341     break;
342
343     case '6':
344         CL;
345         if (data_.size() == 0)
346             createMultisets(true);
347         else
348             createRandom();
349         W;
350         break;
351
352     case '0':
353         return 0;
354
355     default:
356         std::cout << "Invalid option. Please, try again!\n";
357         goto Again;
358     }
359 }
360 }

```

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