



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

ДОМАШНЯЯ РАБОТА №2

«Стандартная библиотека шаблонов STL»

ДИСЦИПЛИНА: «Высокоуровневое программирование»

Выполнил: студент гр. ИУК4-22Б _____ (_____ Карельский М.К.)
(Подпись) (Ф.И.О.)

Проверил: _____ (_____ Козина А.В.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга , 2021

Цель: освоение технологии обобщенного программирования с использованием библиотеки стандартных шаблонов (STL) языка C++.

Задачи:

1. Познакомиться со структурой STL.
2. Изучить основные контейнеры, итераторы, алгоритмы библиотеки STL.
3. Освоить порядок работы со стандартными контейнерами, итераторами и алгоритмами библиотеки STL.

Вариант 20

Задание:

Написать и отладить три программы.

Первая программа демонстрирует использование контейнерных классов для хранения встроенных типов данных.

Вторая программа демонстрирует использование контейнерных классов для хранения пользовательских типов данных.

Третья программа демонстрирует использование алгоритмов STL.

В программе № 1 выполнить следующее:

1. Создать объект первого контейнера **multiset** и заполнить его данными типа **int**.
2. Просмотреть контейнер.
3. Изменить контейнер, удалив из него одни элементы и заменив другие.
4. Просмотреть контейнер, используя для доступа к его элементам итераторы.
5. Создать второй контейнер этого же класса и заполнить его данными того же типа, что и первый контейнер.
6. Изменить первый контейнер, удалив из него n элементов после заданного и добавив затем в него все элементы из второго контейнера.
7. Просмотреть первый и второй контейнеры.

В программе № 2 выполнить то же самое, но для данных пользовательского типа (созданный класс из ЛР №1).

В программе № 3 выполнить следующее:

1. Создать контейнер, содержащий объекты пользовательского типа. Тип контейнера – **vector**.
2. Отсортировать его по убыванию элементов.
3. Просмотреть контейнер.
4. Используя подходящий алгоритм, найти в контейнере элемент, удовлетворяющий заданному условию.
5. Переместить элементы, удовлетворяющие заданному условию в другой (предварительно пустой) контейнер.
6. Просмотреть второй контейнер.
7. Отсортировать первый и второй контейнеры по возрастанию элементов.

8. Просмотреть их.
9. Получить третий контейнер путем слияния первых двух.
- 10.Просмотреть третий контейнер.
- 11.Подсчитать, сколько элементов, удовлетворяющих заданному условию, содержит третий контейнер.
- 12.Определить, есть ли в третьем контейнере элемент, удовлетворяющий заданному условию.

Листинг:

Exam.h:

```
#ifndef EXAM_H
#define EXAM_H

class Exam
{
    char* m_studentName{};
    int m_date{};
    int m_mark{};

public:
    Exam();
    Exam(const char* studentName, const int date, const int mark);
    Exam(const Exam & exam);
    ~Exam();

    char* GetStudentName();
    int GetDate();
    int GetMark();

    int SetStudentName(char* studentName);
    int SetDate(int date);
    int SetMark(int mark);

    void Print() const;

    friend bool operator<(const Exam& exam1, const Exam& exam2);
    friend bool operator>(const Exam& exam1, const Exam& exam2);
    Exam& operator=(const Exam& exam1);
};

#endif
```

Exam.cpp:

```
#include "Exam.h"
#include <iostream>

using namespace std;

const int LENGTH_OF_STUDENT_NAME = 255;

Exam::Exam()
```

```

{
    m_studentName = new char[LENGTH_OF_STUDENT_NAME]{};
    strcpy_s(m_studentName, LENGTH_OF_STUDENT_NAME, "Isaac");
    m_date = 10100;
    m_mark = 1;
}

Exam::Exam(const char* studentName, const int date, const int
mark)
{
    m_studentName = new char[LENGTH_OF_STUDENT_NAME]{};
    strcpy_s(m_studentName, LENGTH_OF_STUDENT_NAME, studentName);
    m_date = date;
    m_mark = mark;
}

Exam::Exam(const Exam & exam)
{
    m_studentName = new char[LENGTH_OF_STUDENT_NAME]{};
    strcpy_s(m_studentName, LENGTH_OF_STUDENT_NAME,
exam.m_studentName);
    m_date = exam.m_date;
    m_mark = exam.m_mark;
}

Exam::~~Exam() { delete[] m_studentName; }

char* Exam::GetStudentName() { return m_studentName; }
int Exam::GetDate() { return m_date; }
int Exam::GetMark() { return m_mark; }

int Exam::SetStudentName(char* studentName)
{
    if (strlen(studentName) < LENGTH_OF_STUDENT_NAME)
    {
        strcpy_s(m_studentName, LENGTH_OF_STUDENT_NAME,
studentName);
        return 0;
    }
    else
    {
        return -1;
    }
}

int Exam::SetDate(int date)
{
    if (date < 0)
    {
        return -1;
    }
    else
    {
        int day = date / 10000;
        int month = (date % 10000) / 100;

```

```

        if (day >= 1 && day <= 31 && month >= 1 && month <= 12)
        {
            m_date = date;
            return 0;
        }
        else
        {
            return -1;
        }
    }
}

int Exam::SetMark(int mark)
{
    if (mark >= 1 && mark <= 5)
    {
        m_mark = mark;
        return 0;
    }
    else
    {
        return -1;
    }
}

void Exam::Print() const
{
    cout << "Student name: " << m_studentName << '\n';
    cout << "Date: " << m_date << '\n';
    cout << "Mark: " << m_mark << '\n';
    cout << '\n';
}

bool operator<(const Exam& exam1, const Exam& exam2) { return
exam1.m_mark < exam2.m_mark; }
bool operator>(const Exam& exam1, const Exam& exam2) { return
exam1.m_mark > exam2.m_mark; }

Exam& Exam::operator=(const Exam& exam)
{
    if (this == &exam)
    {
        return *this;
    }
    else
    {
        strcpy_s(m_studentName, LENGTH_OF_STUDENT_NAME,
exam.m_studentName);
        m_date = exam.m_date;
        m_mark = exam.m_mark;
        return *this;
    }
}

```

Main.cpp:

```
#include <iostream>
#include <set>
#include "Exam.h"
#include <vector>
#include <algorithm>

using namespace std;

void RunFirst()
{
    multiset<int> container1{};

    cout << "Введите 7 целых чисел для первого контейнера\n";
    int input{};
    for (unsigned short i{}; i < 7; ++i)
    {
        cout << i + 1 << ": ";
        cin >> input;
        container1.insert(input);
    }

    cout << "\nСодержимое первого контейнера: ";
    multiset<int>::iterator iterator{};
    for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
    {
        cout << *iterator << " ";
    }
    cout << "\n";

    cout << "\nКакие 2 элемента (по значению) вы хотите
удалить?\n";
    cout << "1: ";
    cin >> input;
    container1.erase(input);
    cout << "2: ";
    cin >> input;
    container1.erase(input);

    cout << "\nКакие 2 элемента (по значению) вы хотите
заменить?\n";
    cout << "1: ";
    cin >> input;
    container1.erase(input);
    cout << "2: ";
    cin >> input;
    container1.erase(input);

    cout << "\nКакие 2 элемента вы хотите записать вместо них?\n";
    cout << "1: ";
    cin >> input;
    container1.insert(input);
    cout << "2: ";
```

```

    cin >> input;
    container1.insert(input);

    cout << "\nСодержимое первого контейнера: ";
    for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
    {
        cout << *iterator << " ";
    }
    cout << "\n";

    multiset<int> container2{};

    cout << "\nВведите 3 целых числа для второго контейнера\n";
    for (unsigned short i{}; i < 3; ++i)
    {
        cout << i + 1 << ": ";
        cin >> input;
        container2.insert(input);
    }

    cout << "\nСколько элементов вы хотите удалить в первом
контейнере?\n";
    cout << ">>> ";
    int length{};
    cin >> length;
    cout << "\nНачиная с какого элемента (по значению) произвести
удаление?\n";
    cout << ">>> ";
    cin >> input;
    iterator = container1.find(input);
    for (int i = 0; i < length; ++i)
    {
        ++iterator;
    }
    container1.erase(container1.find(input), iterator);

    container1.insert(container2.begin(), container2.end());
    cout << "\nСодержимое первого контейнера после объединения со
вторым: ";
    for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
    {
        cout << *iterator << " ";
    }
    cout << "\n";

    cout << "\nСодержимое второго контейнера: ";
    for (iterator = container2.begin(); iterator !=
container2.end(); ++iterator)
    {
        cout << *iterator << " ";
    }
    cout << "\n\n";
}

```

```

void RunSecond()
{
    multiset<Exam> container1{};

    cout << "Введите 4 экзамена для первого контейнера\n\n";
    for (unsigned short i{}; i < 4; ++i)
    {
        Exam exam = Exam();

        cout << "#" << i + 1 << "\n";

        cout << "Имя студента: ";
        char* name = new char[255];
        cin.getline(name, 255, '\n');
        exam.SetStudentName(name);
        delete[] name;

        cout << "Дата экзамена: ";
        int date{};
        cin >> date;
        exam.SetDate(date);

        cout << "Оценка: ";
        int mark{};
        cin >> mark;
        exam.SetMark(mark);
        cout << "\n";
        cin.ignore();

        container1.insert(exam);
    }

    cout << "\nСодержимое первого контейнера\n\n";
    multiset<Exam>::iterator iterator{};
    for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
    {
        (*iterator).Print();
    }

    int input{};
    cout << "\nКакой экзамен вы хотите удалить?\n";
    cout << ">>> ";
    cin >> input;
    iterator = container1.begin();
    for (int i = 0; i < input - 1; ++i)
    {
        ++iterator;
    }
    container1.erase(iterator);

    cout << "\n\nСодержимое первого контейнера\n\n";
    for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)

```



```

{
    (*iterator).Print();
}

cout << "\nКакой экзамен вы хотите изменить?\n";
cout << ">>> ";
cin >> input;
iterator = container1.begin();
for (int i = 0; i < input - 1; ++i)
{
    ++iterator;
}
container1.erase(iterator);

cout << "\nВведите новые данные\n\n";
Exam newExam = Exam();

cin.ignore();
cout << "Имя студента: ";
char* newName = new char[255];
cin.getline(newName, 255, '\n');
newExam.SetStudentName(newName);
delete[] newName;

cout << "Дата экзамена: ";
int newDate{};
cin >> newDate;
newExam.SetDate(newDate);

cout << "Оценка: ";
int newMark{};
cin >> newMark;
newExam.SetMark(newMark);
cout << "\n";

container1.insert(newExam);

cout << "\nСодержимое первого контейнера\n\n";
for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
{
    (*iterator).Print();
}
cout << "\n";

multiset<Exam> container2{};

cout << "Введите 2 экзамена для второго контейнера\n\n";
for (unsigned short i{}; i < 2; ++i)
{
    Exam exam = Exam();

    cout << "#" << i + 1 << "\n";

    cin.ignore();
}

```

```

        cout << "Имя студента: ";
        char* name = new char[255];
        cin.getline(name, 255, '\n');
        exam.SetStudentName(name);
        delete[] name;

        cout << "Дата экзамена: ";
        int date{};
        cin >> date;
        exam.SetDate(date);

        cout << "Оценка: ";
        int mark{};
        cin >> mark;
        exam.SetMark(mark);
        cout << "\n";

        container2.insert(exam);
    }

    cout << "\nСколько экзаменов вы хотите удалить в первом
контейнере?\n";
    cout << ">>> ";
    int length{};
    cin >> length;
    cout << "\nНачиная с какого элемента произвести удаление?\n";
    cout << ">>> ";
    cin >> input;
    cout << "\n";
    iterator = container1.begin();
    for (int i = 0; i < input - 1; ++i)
    {
        ++iterator;
    }
    multiset<Exam>::iterator endIterator = iterator;
    for (int i = 0; i < length; ++i)
    {
        ++endIterator;
    }
    container1.erase(iterator, endIterator);

    container1.insert(container2.begin(), container2.end());
    cout << "\nСодержимое первого контейнера после объединения со
вторым\n\n";
    for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
    {
        (*iterator).Print();
    }

    cout << "\nСодержимое второго контейнера\n\n";
    for (iterator = container2.begin(); iterator !=
container2.end(); ++iterator)
    {
        (*iterator).Print();
    }

```

```

    }
}

bool Compare(Exam& exam1, Exam& exam2)
{
    return exam1 > exam2;
}

bool IsPassed(Exam& exam)
{
    return exam.GetMark() > 2;
}

bool Is1(Exam& exam)
{
    return exam.GetMark() == 1;
}

void RunThird()
{
    vector<Exam> container1{};

    cout << "Введите 6 экзаменов для первого контейнера\n";
    for (unsigned short i{}; i < 6; ++i)
    {
        Exam* exam = new Exam();

        cout << "#" << i + 1 << "\n";

        cout << "Имя студента: ";
        char* name = new char[255];
        cin.getline(name, 255, '\n');
        exam->SetStudentName(name);
        delete[] name;

        cout << "Дата экзамена: ";
        int date{};
        cin >> date;
        exam->SetDate(date);

        cout << "Оценка: ";
        int mark{};
        cin >> mark;
        exam->SetMark(mark);
        cout << "\n";
        cin.ignore();

        container1.push_back(*exam);
    }

    sort(container1.begin(), container1.end(), Compare);

    cout << "\nСодержимое первого контейнера после
сортировки\n\n";
    vector<Exam>::iterator iterator{};

```

```

        for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
        {
            (*iterator).Print();
        }

        cout << "\nПервый сданный экзамен в первом контейнере\n\n";
        find_if(container1.begin(), container1.end(), IsPassed) -
>Print();

        vector<Exam> container2{};
        while (find_if(container1.begin(), container1.end(), IsPassed)
!= container1.end())
        {
            container2.push_back(*find_if(container1.begin(),
container1.end(), IsPassed));
            container1.erase(find_if(container1.begin(),
container1.end(), IsPassed));
        }

        cout << "\nСодержимое второго контейнера после переноса\nв
него всех сданных экзаменов из первого контейнера\n\n";
        for (iterator = container2.begin(); iterator !=
container2.end(); ++iterator)
        {
            (*iterator).Print();
        }

        sort(container1.begin(), container1.end());
        sort(container2.begin(), container2.end());

        cout << "\nСодержимое первого контейнера после
сортировки\n\n";
        for (iterator = container1.begin(); iterator !=
container1.end(); ++iterator)
        {
            (*iterator).Print();
        }

        cout << "\nСодержимое второго контейнера после
сортировки\n\n";
        for (iterator = container2.begin(); iterator !=
container2.end(); ++iterator)
        {
            (*iterator).Print();
        }

        vector<Exam> container3{};
        container3.insert(container3.end(), container1.begin(),
container1.end());
        container3.insert(container3.end(), container2.begin(),
container2.end());

        cout << "\nСодержимое третьего контейнера после слияния
первого и второго\n\n";

```

```

        for (iterator = container3.begin(); iterator !=
container3.end(); ++iterator)
        {
            (*iterator).Print();
        }

        cout << "\nКоличество сданных экзаменов: " <<
count_if(container3.begin(), container3.end(), IsPassed) << "\n";

        if (find_if(container3.begin(), container3.end(), Is1) !=
container3.end())
        {
            cout << "\nЕсть те, кто написал экзамен на 1\n";
        }
        else
        {
            cout << "\nНикто не написал экзамен на 1\n";
        }

        cout << "\n";
    }

int main()
{
    setlocale(LC_ALL, "Russian");

    unsigned short command = 1;
    while (command != 0)
    {
        cout <<
"===== \n\n";
        cout << "Какую задачу запустить?\n";
        cout << "1. Контейнеры со встроенным типом данных\n";
        cout << "2. Контейнеры с пользовательским типом
данных\n";
        cout << "3. Алгоритмы\n";
        cout << "0. Выход\n";
        cout << ">> ";
        cin >> command;
        cout <<
"\n===== \n\n";
        cin.ignore();

        switch (command)
        {
        case 1:
            RunFirst();
            break;
        case 2:
            RunSecond();
            break;
        case 3:
            RunThird();
            break;
        }
    }
}

```

```

    }

    return 0;
}

```

Использованные контейнеры:

Multiset – шаблонный ассоциативный контейнер, хранящий отсортированные элементы, которые могут повторяться.

```

multiset<int> container1{};
multiset<Exam> container1{};

```

Vector – шаблонный последовательный контейнер переменного размера, хранящий последовательность элементов, поддерживает произвольный доступ к любому элементу, добавление и удаление элементов из любого места контейнера.

```

vector<Exam> container1{};

```

Использованные итераторы:

Двунаправленные итераторы мультимножества типа `int` и `Exam`.

```

multiset<int>::iterator iterator{};
multiset<Exam>::iterator iterator{};

```

Итератор произвольного доступа вектора типа `Exam`.

```

vector<Exam>::iterator iterator{};

```

Использованные алгоритмы:

Сортировка элементов, начиная с элемента, на которого указывает итератор начала контейнера, и заканчивая перед элементом, на которого указывает итератор конца контейнера, правило сортировки определено компаратором.

```

sort(container1.begin(), container1.end(), Compare);

```

Сортировка элементов, начиная с элемента, на которого указывает итератор начала контейнера, и заканчивая перед элементом, на которого указывает итератор конца контейнера.

```

sort(container1.begin(), container1.end());

```

Поиск первого элемента, удовлетворяющего условию, которое задано предикатом, в границах от элемента, на которого указывает итератор начала

контейнера, до элемента (не включительно), на которого указывает итератор конца контейнера.

```
find_if(container1.begin(), container1.end(), IsPassed)
```

Подсчет элементов, удовлетворяющих условию, которое определено предикатом, в границах от элемента, на которого указывает итератор начала контейнера, до элемента (не включительно), на которого указывает итератор конца контейнера.

```
count_if(container3.begin(), container3.end(), IsPassed)
```

Результат работы:

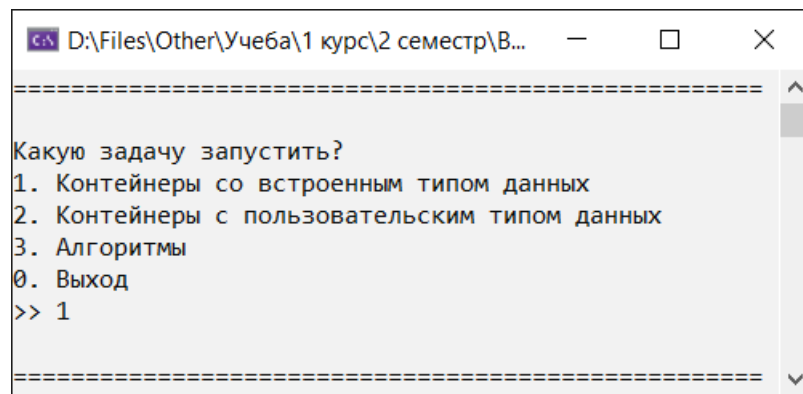


Рисунок 1. Основное меню

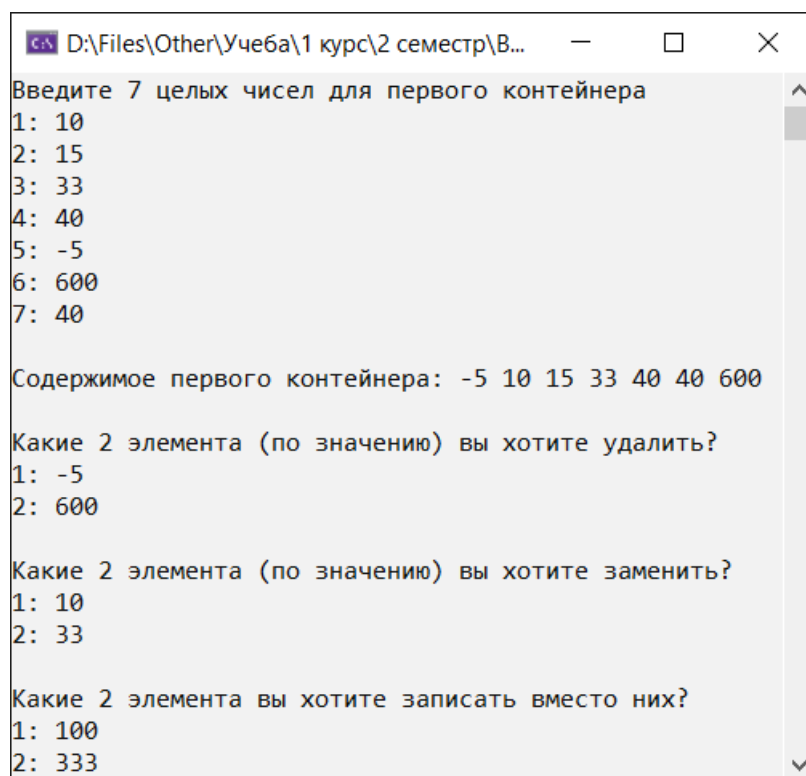
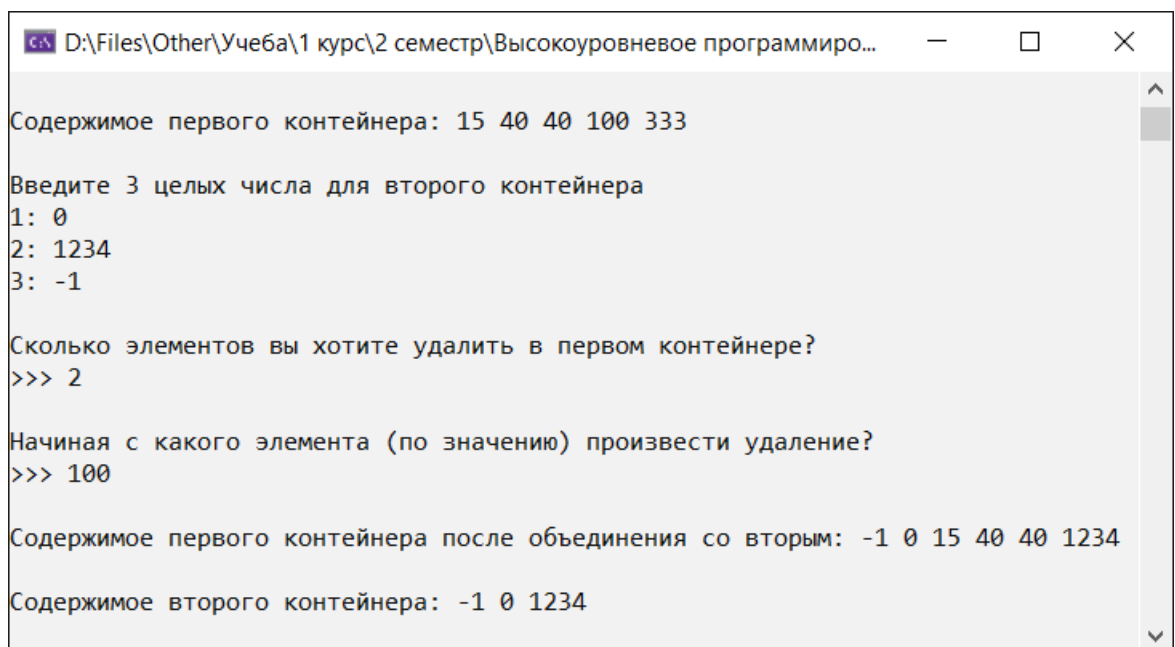


Рисунок 2.1. Первая задача



A screenshot of a Python console window. The title bar shows the file path: D:\Files\Other\Учеба\1 курс\2 семестр\Высокоуровневое программиро... The console output is as follows:

```
Содержимое первого контейнера: 15 40 40 100 333

Введите 3 целых числа для второго контейнера
1: 0
2: 1234
3: -1

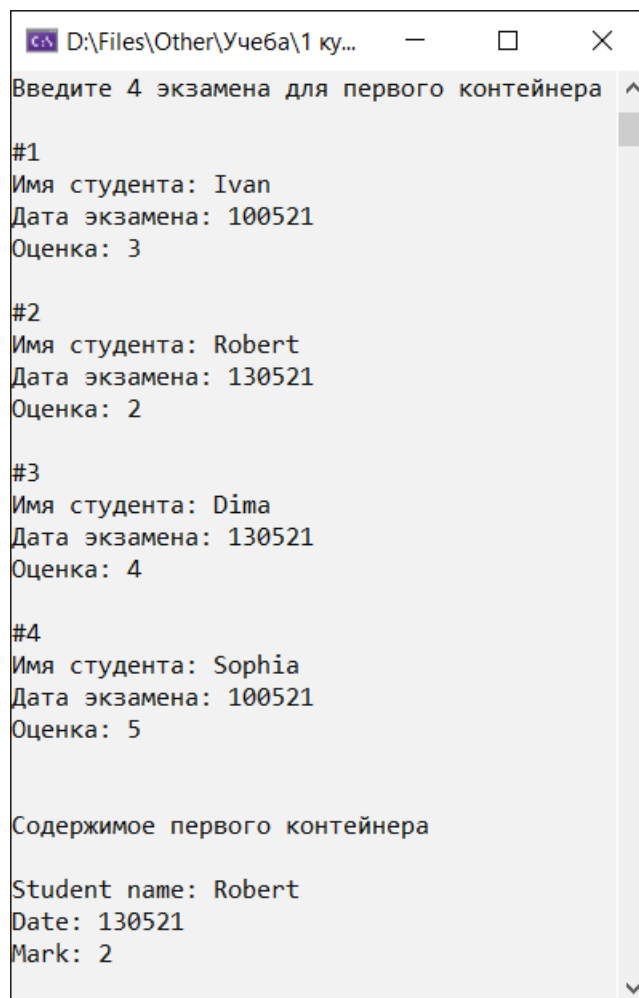
Сколько элементов вы хотите удалить в первом контейнере?
>>> 2

Начиная с какого элемента (по значению) произвести удаление?
>>> 100

Содержимое первого контейнера после объединения со вторым: -1 0 15 40 40 1234

Содержимое второго контейнера: -1 0 1234
```

Рисунок 2.2. Первая задача



A screenshot of a Python console window. The title bar shows the file path: D:\Files\Other\Учеба\1 ку... The console output is as follows:

```
Введите 4 экзамена для первого контейнера

#1
Имя студента: Ivan
Дата экзамена: 100521
Оценка: 3

#2
Имя студента: Robert
Дата экзамена: 130521
Оценка: 2

#3
Имя студента: Dima
Дата экзамена: 130521
Оценка: 4

#4
Имя студента: Sophia
Дата экзамена: 100521
Оценка: 5

Содержимое первого контейнера

Student name: Robert
Date: 130521
Mark: 2
```

Рисунок 3.1. Вторая задача


```
D:\Files\Other\Y...
Student name: Ivan
Date: 100521
Mark: 3

Student name: Dima
Date: 130521
Mark: 4

Student name: Sophia
Date: 100521
Mark: 5

Какой экзамен вы хотите удалить?
>>> 1

Содержимое первого контейнера

Student name: Ivan
Date: 100521
Mark: 3

Student name: Dima
Date: 130521
Mark: 4

Student name: Sophia
Date: 100521
Mark: 5

Какой экзамен вы хотите изменить?
>>> 1

Введите новые данные

Имя студента: Ivan
Дата экзамена: 100521
Оценка: 4
```

Рисунок 3.2. Вторая задача

```
D:\Files\Other\Учеба\1 курс\2 семестр\Высоко...
Содержимое первого контейнера
Student name: Dima
Date: 130521
Mark: 4

Student name: Ivan
Date: 100521
Mark: 4

Student name: Sophia
Date: 100521
Mark: 5

Введите 2 экзамена для второго контейнера
#1
Имя студента: Artem
Дата экзамена: 130521
Оценка: 4

#2
Имя студента: Nikita
Дата экзамена: 160521
Оценка: 5

Сколько экзаменов вы хотите удалить в первом контейнере?
>>> 2

Начиная с какого элемента произвести удаление?
>>> 2

Содержимое первого контейнера после объединения со вторым
Student name: Dima
Date: 130521
Mark: 4

Student name: Artem
Date: 130521
Mark: 4
```

Рисунок 3.3. Вторая задача

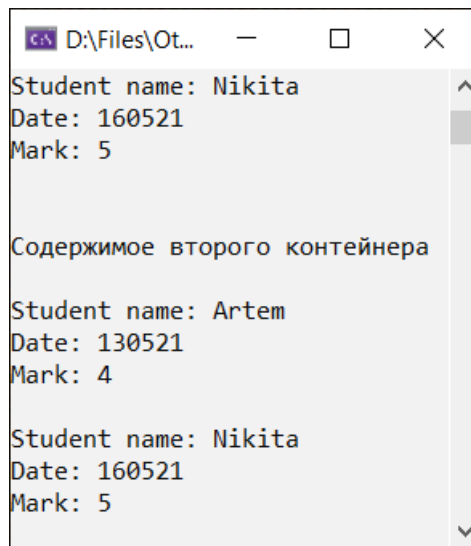


Рисунок 3.4. Вторая задача

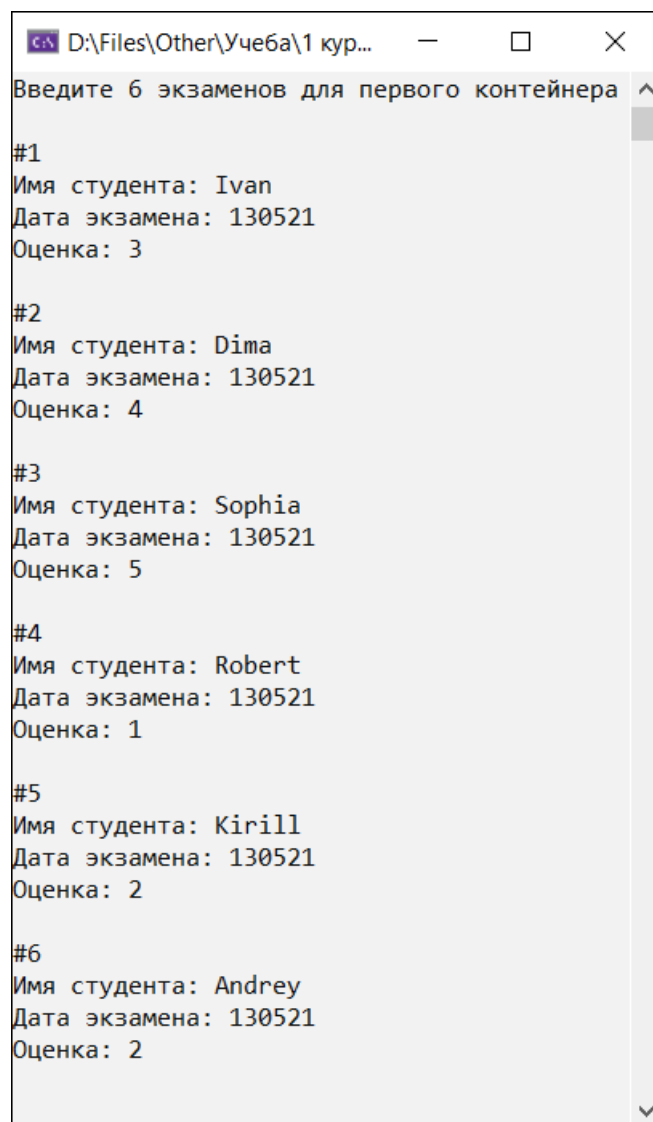


Рисунок 4.1. Третья задача

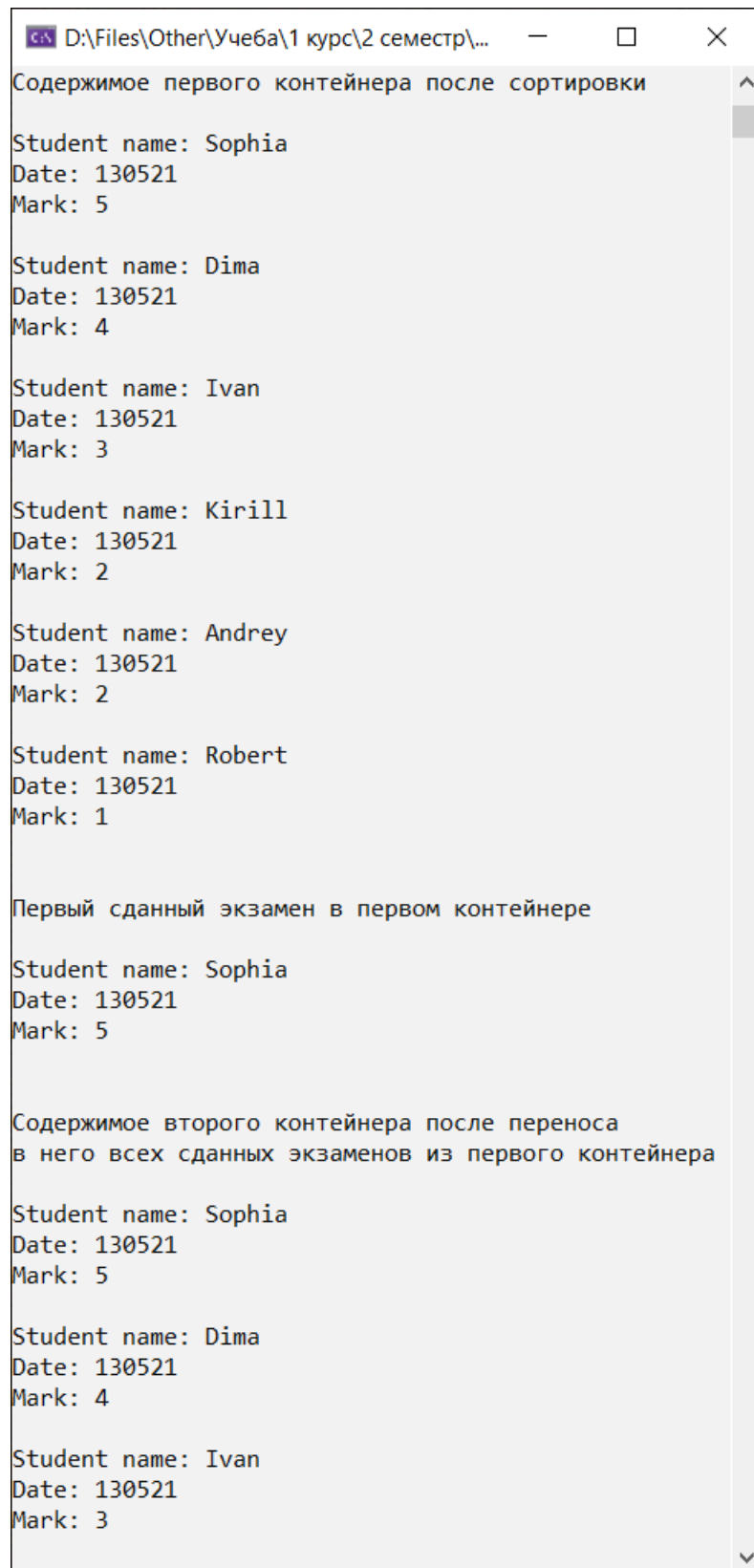


Рисунок 4.2. Третья задача

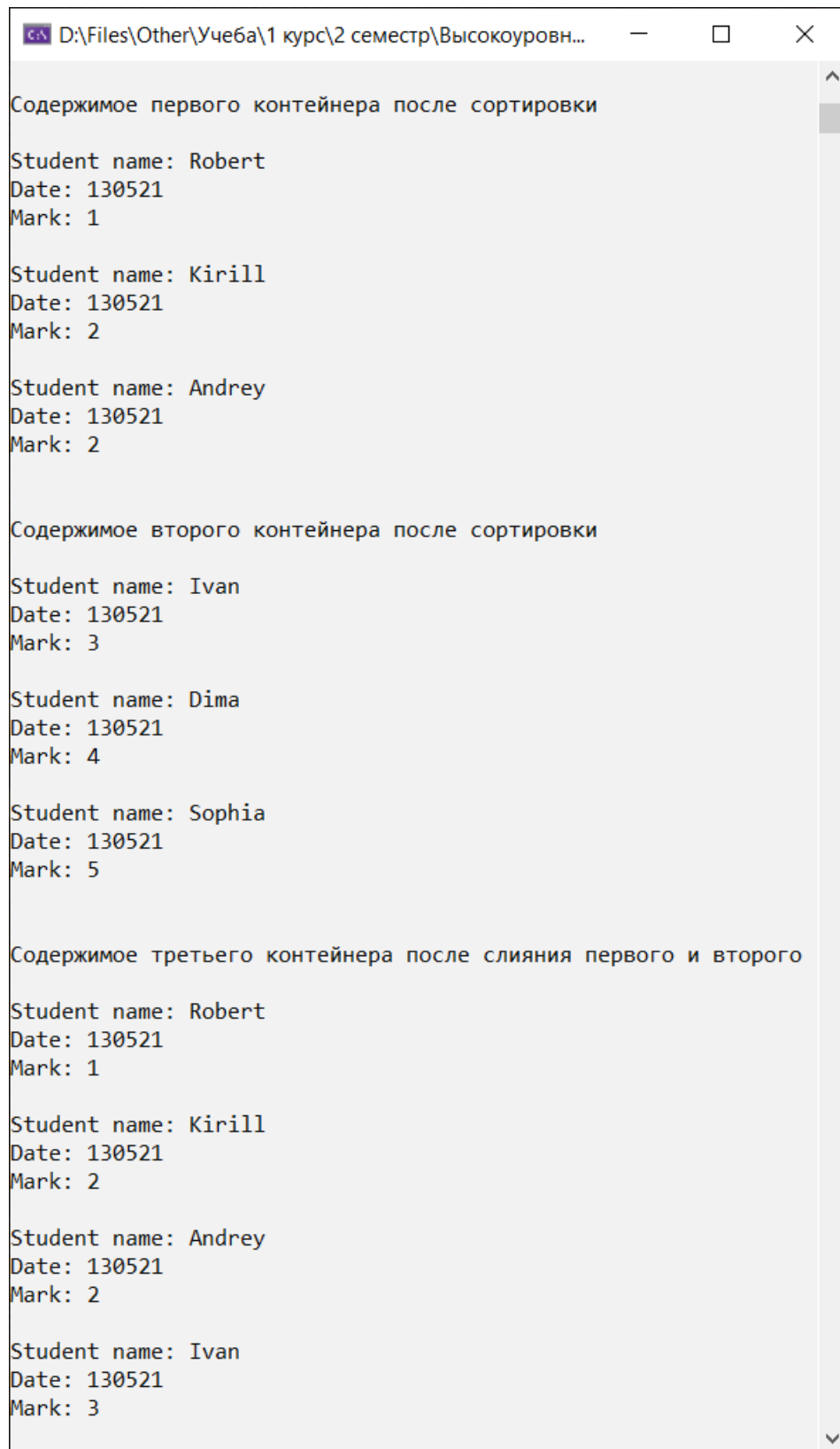
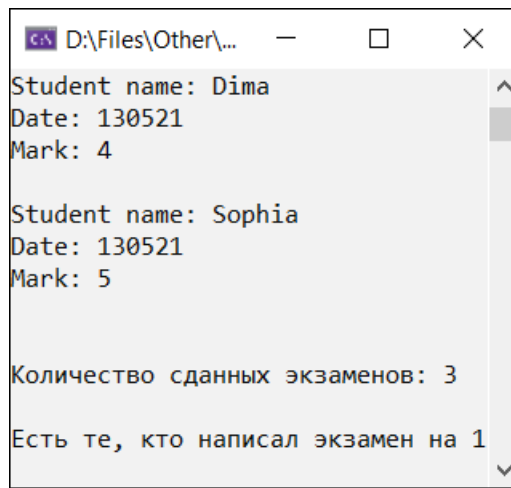


Рисунок 4.3. Третья задача



```

D:\Files\Other\...
Student name: Dima
Date: 130521
Mark: 4

Student name: Sophia
Date: 130521
Mark: 5

Количество сданных экзаменов: 3

Есть те, кто написал экзамен на 1

```

Рисунок 4.4. Третья задача

Вывод: в ходе выполнения работы были получены практические навыки работы с шаблонными контейнерами `multiset` и `vector`, хранящими встроенный и пользовательский типы данных, итераторами, шаблонными функциями сортировки, поиска элемента и вычисления количества искомых элементов, создания предикатов.