ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа

Стандартные обобщённые алгоритмы библиотеки STL

Выполнил студент группы РИС-23-3Б

Мазунин М.А.

Проверила доцент кафедры ИТАС

О. А. Полякова

2024 г.

**1. Постановка задачи**

Задача 1.

1. Создать последовательный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).
3. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
4. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 2.

1. Создать адаптер контейнера.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
3. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
4. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 3

1. Создать ассоциативный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
3. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
4. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**2. Анализ задачи**

Класс Pair - Класс, содержащий пару чисел

Класс List - Класс-контейнер список из библиотеки STL

Класс Priority\_Queue - Класс-контейнер очередь с приоритетами из библиотеки STL

Класс PrioCondition - Класс, содержащий условие приоритета для класса Priority\_Queue

Класс Map - Класс-контейнер словарь из библиотеки STL

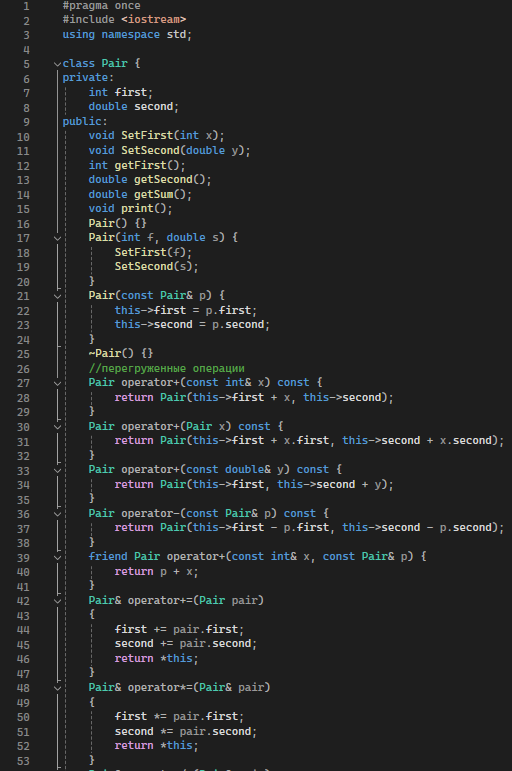
**3. Разбор работы алгоритма**

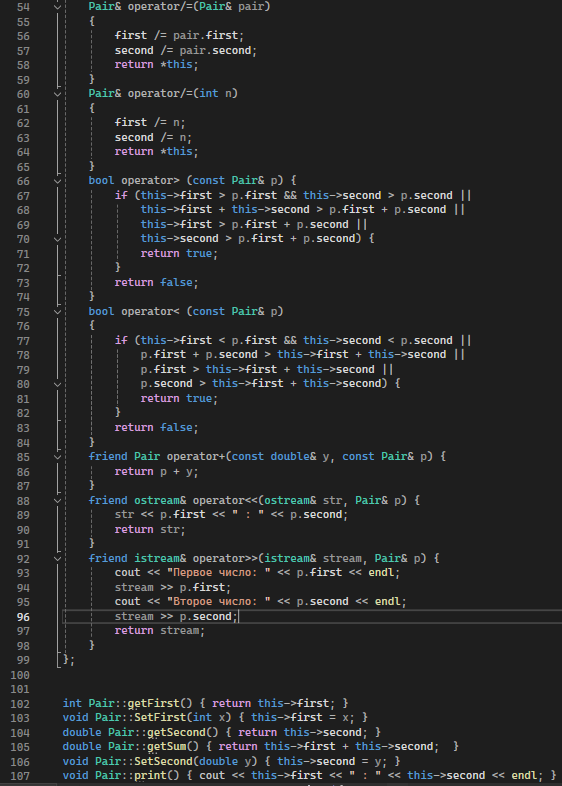
UML диаграмма



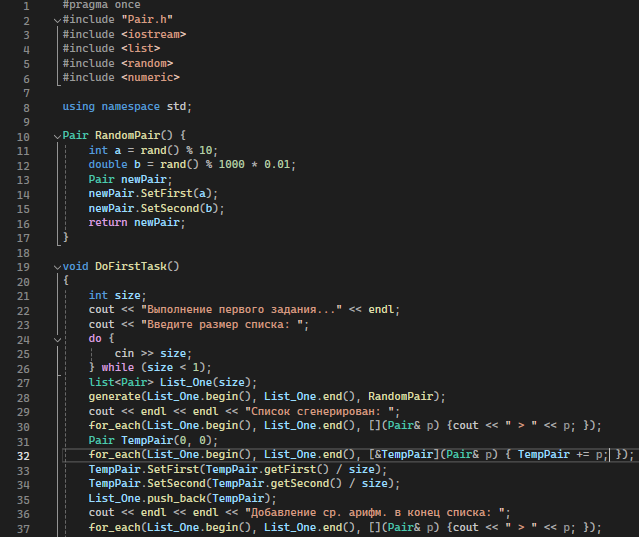
Код программы

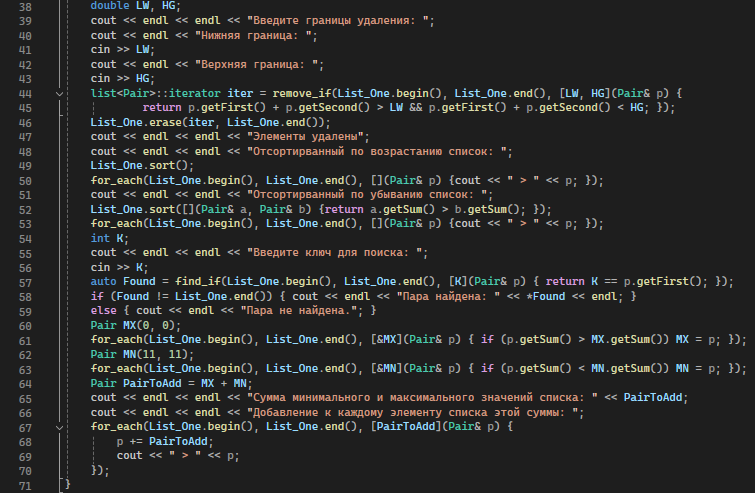
Pair.h





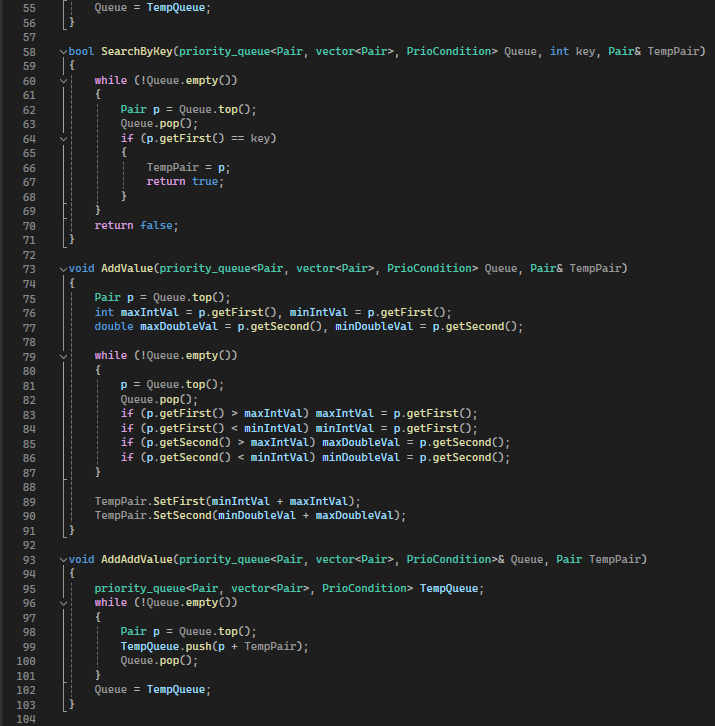
First\_Task.h

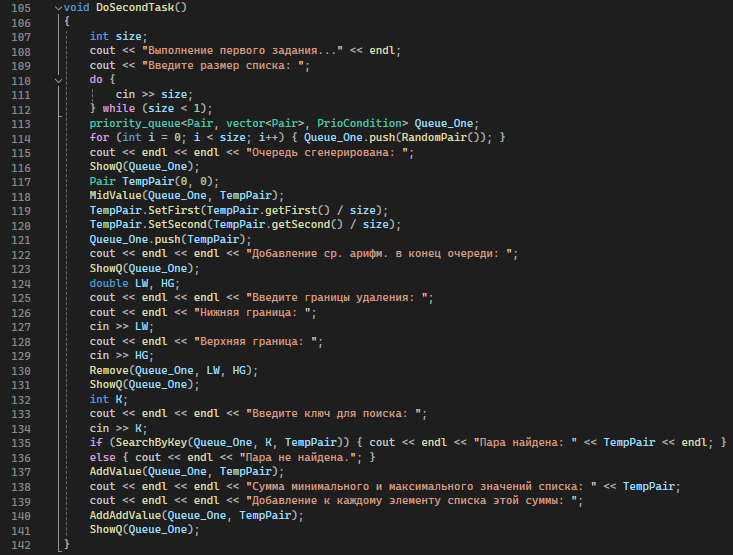




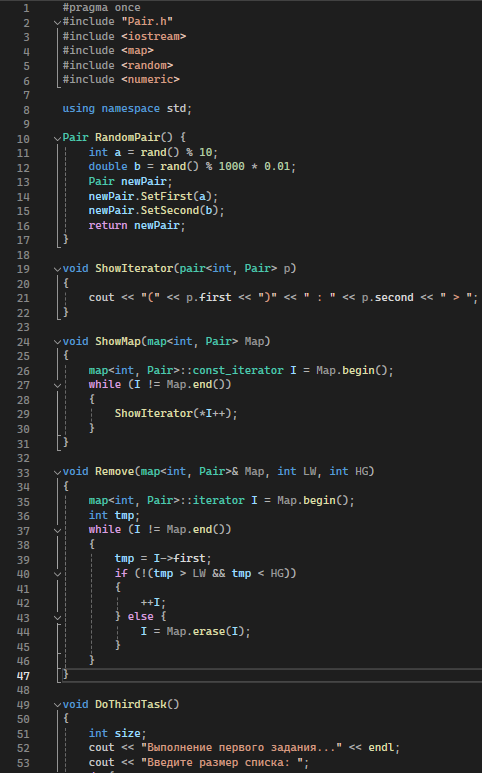
Second\_Task.h

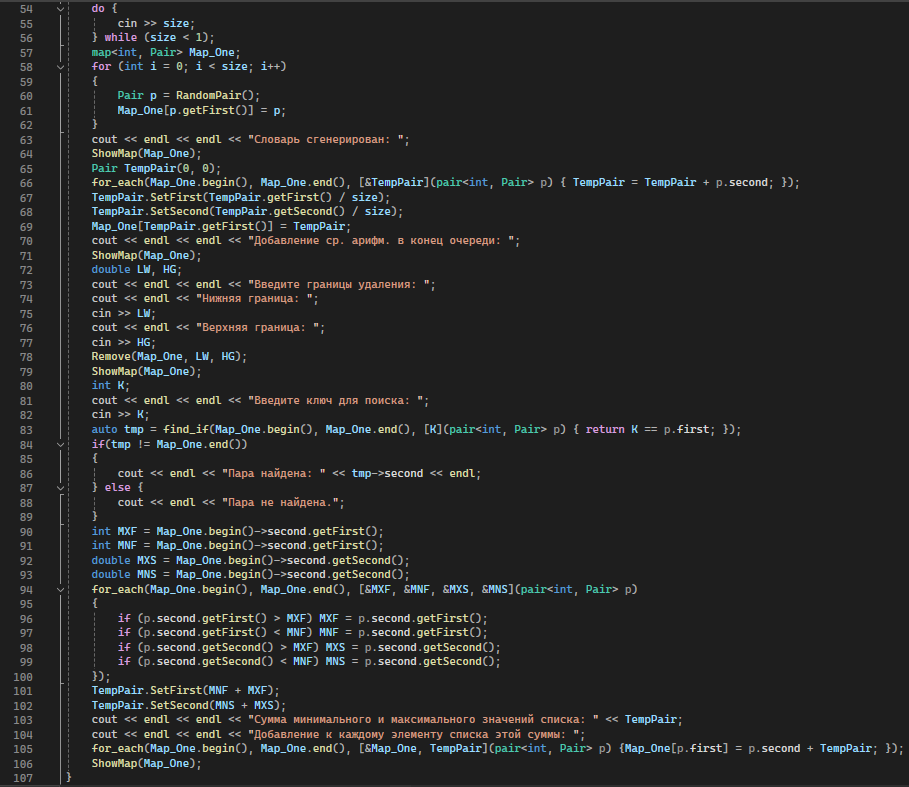




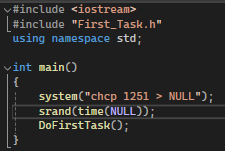


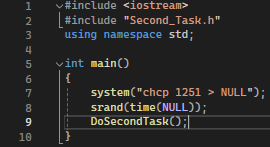
Third\_Task.h

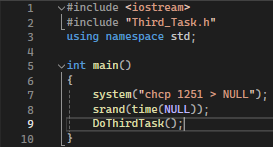




Class\_13.cpp (Первое, второе, третье задание)

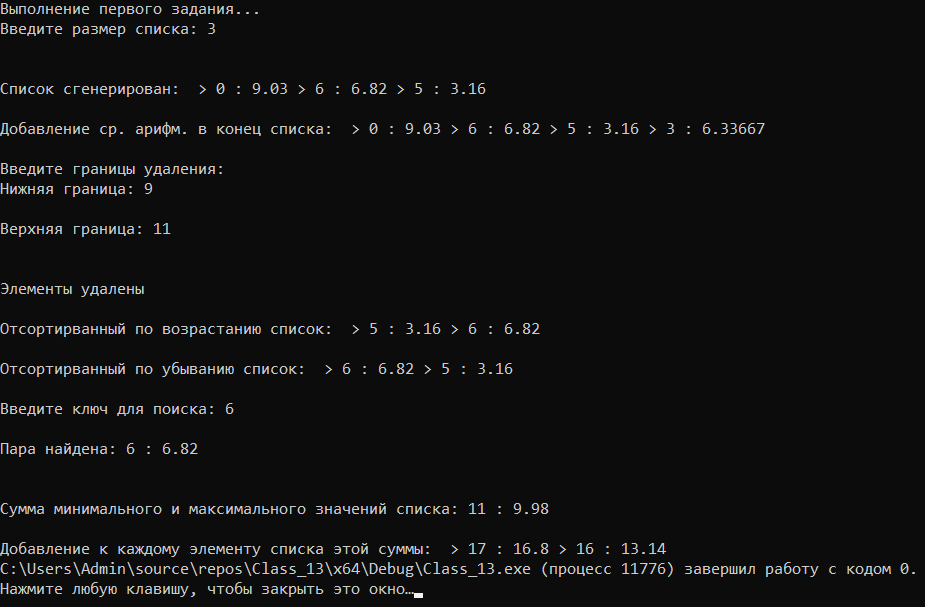




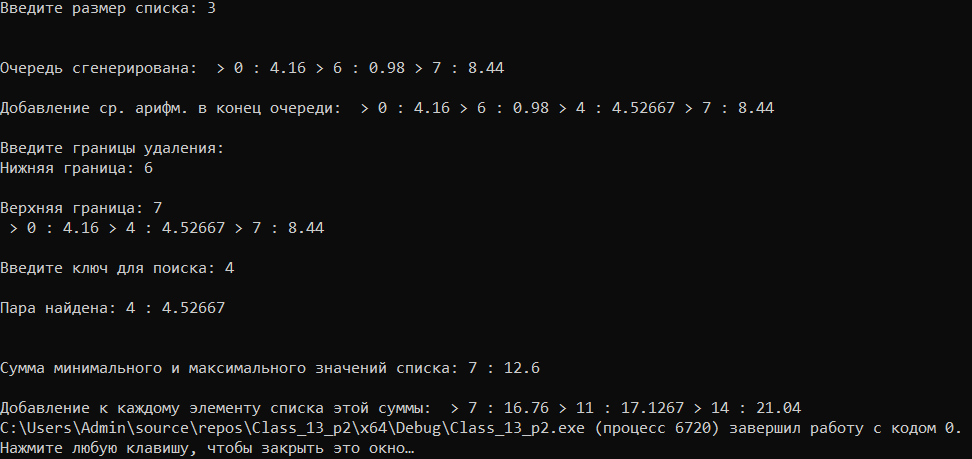


**4. Результаты работы программы**

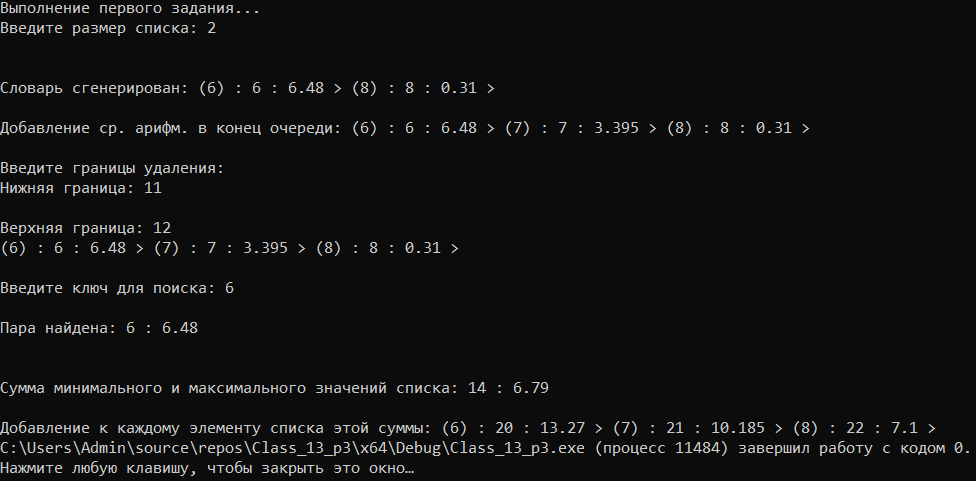
Первое задание



Второе задание



Третье задание



**5. Контрольные вопросы**

ЛР №11

1. Набор контейнерных классов и набор обобщённых алгоритмов
2. Вектор, список, стек, очередь, двунаправленная очередь, очередь с приоритетами, словарь, словарь с дубликатами, множество, множество с дубликатами
3. Подключить соответствующий заголовочный файл
4. Итератор - класс, используемый для поиска элементов в контейнере
5. Разыменовывание, присваивание, сравнение, перемещение по всем элементам контейнера
6. for (i = first; i != last; ++i)
7. входные, выходные, прямые, двунаправленные, произвольного доступа
8. == / !=; ==; clear; insert; erase; size\_type size() const; size\_type max\_size() const; bool empty() const; iterator begin(), iterator end(); reverse\_iterator begin(); reverse\_iterator end()
9. Вставка/Удаление значения с конца, иначе придётся сдвигать все значения путём копирования
10. Вставка и удаление произвольного поля
11. Вставка и удаление первого и последнего элемента
12. pop\_back(); push\_back(); insert; erase; [] / at; swap; clear()
13. pop\_back(); push\_back(T&key); pop\_front(); push\_front(T&key); insert; erase; swap; clear(); splice
14. pop\_back(); push\_back(T&key); pop\_front(); push\_front(T&key); insert; erase; [] / at
15. Записать все элементы после пятого, удалить с конца все элемент кроме первого, добавить все записанные элементы. Удаление из произвольного места возможно, но не эффективно.
16. v.pop\_back()
17. Записать первый элемент, удалить первые пять элементов с начала, добавить записанный элемент в начало
18. l.pop\_back()
19. Установить итератор на позицию 2, удалить четыре элемента
20. Установить итератор на последнюю позицию. Удалить элемент
21. for (i = first; i != last; i++) { cout << cont[i]; }
22. Контейнеры, реализованные на основе других классов
23. Первый стек инициализирован на базе двусторонней очереди, а второй на базе списка
24. push(); pop(); top(); empty(); size()
25. push(); pop(); front(); back(); empty(); size()
26. В очереди с приоритетами для извлечения выбирается максимальные элемент из хранимых в контейнере
27. Записать все элементы до заданного номера в другой стек, удалить элемент, записать элементы из второго стека в первый
28. Записать все элементы до заданного номера спереди или сзади в другую очередь, удалить элемент, записать элементы обратно в первую очередь
29. for (i = first; i != last; i++) { s.top(); s.pop(); }
30. for (i = first; i != last; i++) { q.front(); q.pop\_front(); }

ЛР №12

1. Ассоциативные контейнеры содержат пары значений: ключ и данные
2. Словарь, словарь с дубликатами, множество, множество с дубликатами
3. С помощью итераторов
4. bool empty(); size\_type size() const; size\_type max\_size() const; insert(); erase(); clear(); swap(); key\_comp(); value\_comp(); find(); count(); lower\_bound(); upper\_bound()
5. explicit map(const Comp& c = Comp(), const Allocator& a = Allocator());
6. Упорядоченные по ключу
7. []; =; ==; <=; >=; !=; <; >;
8. m.insert(make\_pair(a, b))
9. m[i] = pair;
10. for (i = first; i != last; i++) { cout << m[i] }
11. for (i = first; i != last; i++) { cout << m[i] }
12. multimap допускает хранение элементов с одинаковыми ключами
13. В множествах значения не играют роли, отслеживаются только ключи
14. В множествах, в отличие от словарей, значения не играют роли, отслеживаются только ключи
15. set<T> set1;
16. Упорядоченные по ключу
17. insert(); erase(); count();
18. s.insert(a);
19. for (i = set.begin(); i != set.end(); i++) { cout << \*i; }
20. multiset допускает хранение элементов с одинаковыми ключами

**6. Вывод**

Алгоритм выполняет поставленную задачу.

Данный проект можно найти на GitHub



