Лабораторная работа № 05

Тема: Итераторы и аллокаторы

Цель:

- Изучение устройства коллекций в стандартной библиотеке
- Получение навыков в использовании концепции «итератор»
- Получение навыков в использовании концепции «аллокатор»

Порядок выполнения работы

1. Ознакомиться с теоретическим материалом.
2. Получить у преподавателя вариант задания.
3. Реализовать задание своего варианта в соответствии с поставленными требованиями.
4. Подготовить тестовые наборы данных.
5. Создать репозиторий на GitHub.
6. Отправить файлы лабораторной работы в репозиторий.
7. Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы преподавателя (как из числа контрольных, так и по реализации программы).

Требования к программе

- Реализовать наследник std::pmr::memory_resource который реализует стратегию работы с памятью согласно варианта задания.
- memory_resource должен позволять переиспользовать освобождаемую память (освобожденную ранее с помощью do_deallocate)
- memory_resource при уничтожении должен подчищать всю неосвобожденную ранее память
- Реализовать шаблонный контейнер согласно варианту задания, который использует созданный memory_resource через шаблон std::pmr::polymorphic_allocator.
- Реализовать итератор к созданному контейнеру. Итератор должен соответствовать std::forward_iterator_tag.
- Создать код, демонстрирующий работ контейнера с простыми (int) и сложными типами (struct с несколькими полями)

Стратегии работы memory_resource:

1. Фиксированный блок памяти (выделяется один раз), информация о выделенных блоках памяти хранится в std::list
2. Фиксированный блок памяти (выделяется один раз), информация о выделенных блоках памяти хранится в std::vector
3. Фиксированный блок памяти (выделяется один раз), информация о выделенных блоках памяти хранится в std::map
4. Динамическое выделение памяти: для каждого объекта выделяется блок памяти на куче, информация о выделенных блоках сохраняется в std::list
5. Динамическое выделение памяти: для каждого объекта выделяется блок памяти на куче, информация о выделенных блоках сохраняется в std::vector
6. Динамическое выделение памяти: для каждого объекта выделяется блок памяти на куче, информация о выделенных блоках сохраняется в std::map

Варианты заданий:

Вариант	Контейнер	Стратегия memory resource
1.	Динамический массив	1
2.	Стек	2
3.	Однонаправленный список	3
4.	Двунаправленный список	4
5.	Очередь	5
6.	Динамический массив	6
7.	Стек	1
8.	Однонаправленный список	2
9.	Двунаправленный список	3
10.	Очередь	4
11.	Динамический массив	5
12.	Стек	6
13.	Однонаправленный список	1
14.	Двунаправленный список	2
15.	Очередь	3
16.	Динамический массив	4
17.	Стек	5
18.	Однонаправленный список	6
19.	Двунаправленный список	1
20.	Очередь	2