Лабораторная работа № 05
Тема: Основы работы с коллекциями: итераторы

Цель:

- Изучение основ работы с коллекциями, знакомство с шаблоном проектирования «Итератор»;

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

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться **oop_exercise_05** (в случае использования Windows **oop_exercise_05.exe)**

Необходимо зарегистрироваться на GitHub (если студент уже имеет регистрацию на GitHub то можно использовать ее) и создать репозитарий для задания лабораторной работы.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github. Имя репозитория должно быть https://github.com/login/oop_exercise_05

Где login – логин, выбранный студентом для своего репозитория на Github.

Репозиторий должен содержать файлы:

- main.cpp //файл с заданием работы
- CMakeLists.txt // файл с конфигураций CMake
- report.doc // отчет о лабораторной работе

Разработать шаблоны классов согласно варианту задания. Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат. Классы должны иметь публичные поля. Фигуры являются фигурами вращения, т.е. равносторонними (кроме трапеции и прямоугольника). Для хранения координат фигур необходимо использовать шаблон std::pair.

Например:

```
template <class T>
struct Square{
    using vertex_t = std::pair<T,T>;
    vertex_t a,b,c,d;
};
```

Создать шаблон динамической коллекцию, согласно варианту задания:

1. Коллекция должна быть реализована с помощью умных указателей (std::shared_ptr, std::weak_ptr). Опционально использование std::unique_ptr;
2. В качестве параметра шаблона коллекция должна принимать тип данных - фигуры;
3. Реализовать forward_iterator по коллекции;
4. Коллекция должны возвращать итераторы begin() и end();
5. Коллекция должна содержать метод вставки на позицию итератора insert(iterator);
6. Коллекция должна содержать метод удаления из позиции итератора erase(iterator);
7. При выполнении недопустимых операций (например выход аз границы коллекции или удаление не существующего элемента) необходимо генерировать исключения;
8. Итератор должен быть совместим со стандартными алгоритмами (например, std::count_if)
9. Коллекция должна содержать метод доступа:
    - Стек – pop, push, top;
    - Очередь – pop, push, top;
    - Список, Динамический массив – доступ к элементу по оператору [];
10. Реализовать программу, которая:
    - Позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию;
    - Позволяет удалять элемент из коллекции по номеру элемента;
    - Выводит на экран введенные фигуры с помощью std::for_each;
    - Выводит на экран количество объектов, у которых площадь меньше заданной (с помощью std::count_if);

Варианты заданий (выпуклые равносторонние фигуры вращения):

Вариант	**Фигура**	**Контейнер**
1.	Треугольник	Стек
2.	Квадрат	Стек
3.	Прямоугольник	Стек
4.	Трапеция	Стек
5.	Ромб	Стек
6.	5-угольник	Стек
7.	6-угольник	Стек

8.	8-угольник	Стек
9.	Треугольник	Список
10.	Квадрат	Список
11.	Прямоугольник	Список
12.	Трапеция	Список
13.	Ромб	Список
14.	5-угольник	Список
15.	6-угольник	Список
16.	8-угольник	Список
17.	Треугольник	Очередь
18.	Квадрат	Очередь
19.	Прямоугольник	Очередь
20.	Трапеция	Очередь
21.	Ромб	Очередь
22.	5-угольник	Очередь
23.	6-угольник	Очередь
24.	8-угольник	Очередь
25.	Треугольник	Динамический массив
26.	Квадрат	Динамический массив

27.	Прямоугольник	Динамический массив
28.	Трапеция	Динамический массив
29.	Ромб	Динамический массив
30.	5-угольник	Динамический массив
31.	6-угольник	Динамический массив
32.	8-угольник	Динамический массив

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases.
4. Результаты выполнения тестов.
5. Объяснение результатов работы программы.