Лабораторная работа № 06
Тема: Основы работы с коллекциями: аллокаторы

Цель:

- Изучение основ работы с контейнерами, знакомство концепцией аллокаторов памяти;

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_06** (в случае использования Windows **oop_exercise_06.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_06

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. равносторонние (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;
3. Коллекция должна содержать метод доступа:
    - Стек – pop, push, top;
    - Очередь – pop, push, top;
    - Список, Динамический массив – доступ к элементу по оператору [];
4. Реализовать аллокатор, который выделяет фиксированный размер памяти (количество блоков памяти – является параметром шаблона аллокатора). Внутри аллокатор должен хранить указатель на используемый блок памяти и динамическую коллекцию указателей на свободные блоки. Динамическая коллекция должна соответствовать варианту задания (Динамический массив, Список, Стек, Очередь);
5. Коллекция должна использовать аллокатор для выделения и освобождения памяти для своих элементов.
6. Аллокатор должен быть совместим с контейнерами std::map и std::list (опционально – vector).
7. Реализовать программу, которая:
    - Позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию использующую аллокатор;
    - Позволяет удалять элемент из коллекции по номеру элемента;
    - Выводит на экран введенные фигуры с помощью std::for_each;

Варианты заданий (выпуклые равносторонние фигуры вращения):

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

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

25.	Треугольник	Динамический массив	Динамический массив
26.	Квадрат	Динамический массив	Список
27.	Прямоугольник	Динамический массив	Стек
28.	Трапеция	Динамический массив	Очередь
29.	Ромб	Динамический массив	Динамический массив
30.	5-угольник	Динамический массив	Список
31.	6-угольник	Динамический массив	Стек
32.	8-угольник	Динамический массив	Очередь

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.