Лабораторная работа № 01

Тема: Первая программа на C++

Цель:

- Изучить систему сборки CMake
- Изучить базовые операторы и конструкции C++
- Изучить библиотеку для написания Unit-тестов Google Test
- Научится писать простые программы, использующие ввод/вывод через потоки std::cin std::cout

Порядок выполнения работы

- Ознакомиться с теоретическим материалом.
- Получить у преподавателя вариант задания.
- Реализовать задание своего варианта в соответствии с поставленными требованиями.
- Написать Unit-тесты с использованием Google Test.
- Создать репозиторий на GitHub.
- Отправить файлы лабораторной работы в репозиторий.
- 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 (std::cin) и выводить данные в стандартный вывод (std::cout).

Необходимо зарегистрироваться на GitHub и создать репозитарий для задания лабораторных работ.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github (через lms). Необходимо реализовать функцию согласно варианту задания. Функция должна быть помещена в отдельный файл (cpp) и вызываться как из основной программы, так и из тестов.

Варианты заданий

№	ЗАДАНИЕ
1	Вам даны два числа a и b, где 0 ≤ a ≤ b. Представьте, что вы построили последовательность из всех целых чисел от a до b включительно. Требуется подсчитать количество 1 в двоичном представлении всех чисел последовательности. **Пример** Для a = 2 и b = 7 на выходе должно получиться 11 При a = 2 и b = 7 массив имеет вид: [2, 3, 4, 5, 6, 7]. Переведя числа в двоичный формат, получим [10, 11, 100, 101, 110, 111], который содержит 1 + 2 + 1 + 2 + 2 + 3 = 11.
2	У меня сумасшедшее психическое заболевание. Я очень не люблю цифры. Но при этом немного запутанно: Число, которого я боюсь, зависит от того, какой сегодня день недели... Вот конкретное описание моей психической болезни:

	Понедельник --> 12 Вторник --> числа больше 95 Среда --> 34 Четверг --> 0 Пятница --> числа, кратные 2 Суббота --> 56 Воскресенье --> 666 или -666 Напишите функцию, которая принимает строку (день недели) и целое число (проверяемое число), чтобы она сообщала врачу, боюсь я или нет. (возвращает булево число)
3	Напишите функцию, которая принимает строку круглых скобок и определяет, является ли порядок скобок правильным. Функция должна возвращать true, если строка допустима, и false, если недопустима. **Примеры** "()" => true ")(()))" => false "(" => false "(())(((()))(()))" => true Ограничения 0 <= длина ввода <= 100 Все входные данные будут строками, состоящими только из символов (и). Пустые строки считаются сбалансированными (и, следовательно, валидными) и будут проверяться.
4	Функция возвращает количество гласных букв в заданной строке (на английскрм). Входная строка будет состоять только из строчных букв и/или пробелов.
5	Преобразование 12-часового времени, например "8:30 am" или "8:30 pm", в 24-часовое (например, "0830" или "2030") Вам необходимо определить функцию, на вход которой будут поданы час (всегда в диапазоне от 1 до 12, включительно), минута (всегда в диапазоне от 0 до 59, включительно) и период (либо "am", либо "pm"). Ваша задача - вернуть четырехзначную строку, кодирующую это время в 24-часовом формате. **Примечания** По условию, полдень - это 12:00 am, а полночь - 12:00 pm. В 12-часовых часах нет часа 0, и время сразу после полуночи обозначается, например, как 12:15 ночи. В 24-часовых часах это означает 0015.
6	Тролли атакуют ваш раздел комментариев! Обычный способ решения этой ситуации - удаление всех гласных из комментариев троллей, что нейтрализует угрозу. Ваша задача - написать функцию, которая принимает строку и возвращает новую строку с удаленными гласными.

	Например, строка "This website is for losers LOL!" превратится в "Ths wbst s fr lsrs LL!".
7	Ваша задача - разделить шоколадную плитку заданного размера n x m на маленькие квадратики. Каждый квадрат имеет размер 1x1 и является неразрывным. Реализуйте функцию, которая будет возвращать минимальное количество разбиений. Например, если вам дана шоколадка размером 2 x 1, то вы можете разделить ее на отдельные квадратики за один надлом, а для размера 3 x 1 необходимо сделать два перерыва. Если входные данные недопустимы, то следует вернуть 0 (т.е. если у нас нет шоколада для разбиения, то разбиения не требуется). Входные данные всегда будут целым неотрицательным числом.
8	Вам дана строка, представляющая собой число в двоичном виде. Ваша задача - удалить все нулевые биты в этой строке и вернуть соответствующее число (сохранив только "1"). На практике вы должны реализовать эту функцию: long eliminate_unset_bits(string number); **Примеры** eliminate_unset_bits("1101010101010101") -> 255 (= 11111111) eliminate_unset_bits("111") -> 7 eliminate_unset_bits("1000000") -> 1 eliminate_unset_bits("000") -> 0
9	О, нет! Число перепуталось с текстом. Ваша задача - извлечь число из текста. Вам дается строка, состоящая из цифр и букв, перемешанных между собой, и вы должны напечатать все числа в этой строке в том порядке, в котором они встречаются.
10	Напишите функцию, возвращающую строку, каждый элемент которого представляет один бит 32-битного целого числа таким образом, чтобы при выводе на печать он выглядел как двоичное представление целого числа (младший значащий бит справа). Например, 1 = 00000000000000000000000000000001 Присвоить элементу массива 1 или 0 в зависимости от того, является ли бит в соответствующей позиции 1 или 0. Например (вход -> выход): 1 -> [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1] Функция принимает один аргумент (n), который представляет собой положительное целое число, подлежащее переводу в двоичный формат.
11	Пара целых чисел (m, n), таких, что 10 > m > n > 0, (меньше 10), что их сумма, (m + n), и остаток, (m - n), являются совершенными квадратами, равна (5, 4). То есть: $5 + 4 = 9 = 3^2$ $5 - 4 = 1 = 1^2$ (10 > 5 > 4 > 0)

	Парой чисел (m, n), ближайших к 50 и меньших его, обладающих описанным выше свойством, является (45, 36). 45 + 36 = 81 = 9² 45 - 36 = 9 = 3² (50 > 45 > 36 > 0) С помощью функции closest_pair_tonum(), принимающей в качестве верхней границы число upper_limit, мы должны получить ближайшую пару, удовлетворяющую описанному выше свойству. Функция должна возвращать наибольшую пару чисел (m, n), удовлетворяющую верхнему_лимиту > m > n > 0. Заметим, что мы говорим, что пара A (a0, a1) больше пары B (b0, b1), если a0 > b0 или a0 == b0 и a1 > b1. Рассмотрим некоторые случаи: closest_pair_tonum(10) == (5, 4) # (m = 5, n = 4) closest_pair_tonum(30) == (29, 20) closest_pair_tonum(50) == (45, 36)
12	Ваша компания, Timaty Pizza, является вторым по величине интернет-магазином замороженной пиццы. Вы владеете несколькими международными складами, которые используются для хранения замороженной пиццы, и вам необходимо определить, сколько ящиков пиццы вы можете хранить на каждом из них. Компания Timaty недавно стандартизировала свои контейнеры для хранения: все пиццы помещаются в кубические ящики со стороной 16 дюймов. Ящики очень прочные, поэтому их можно укладывать как угодно высоко. Напишите функцию box_capacity(), которая определяет, сколько ящиков можно хранить на данном складе. Функция должна принимать три аргумента: длину, ширину и высоту склада (в футах) и возвращать целое число, представляющее количество ящиков, которые можно хранить на этом пространстве. Например: склад длиной 32 фута, шириной 64 фута и высотой 16 футов может вместить 13 824 ящика, поскольку в нем можно разместить 24 ящика в поперечнике, 48 ящиков в глубину и 12 ящиков в высоту, поэтому box_capacity(32, 64, 16) должна возвращать 13824.
13	Определение Чистым числом называется число, цифры которого расположены в неубывающем порядке. Задача Задано число, найдите, является ли оно чистым или нет.
14	Задача Напишите метод, который заменяет каждый n-ый симовл oldValue на newValue. Входные данные text: строка для модификации

	n: номер целевой буквы old_value : целевой символ new_value : символ, который следует использовать в качестве замены Правила Если n равно 0 или отрицательно, или если оно больше, чем счетчик oldValue, вернуть исходный текст без изменений. **Пример**: n: 2 old_value: 'a' new_value: 'o' "Vader said: No, I am your father!" -> "Vader soid: No, I am your fother!"
15	Задача По делителю и границе, найдите наибольшее целое число N, такое, что , N делится на делитель. N меньше или равно границе N больше 0. Примечания Параметры (devider, bound), передаваемые в функцию, являются только положительными величинами . . **Примеры** ввода >> вывода devider = 2, bound = 7 ==> return (6) Пояснение: (6) делится на (2) , (6) меньше или равно bound (7) , и (6) > 0 . devider = 10, bound = 50 ==> return (50) Пояснение: (50) делится на (10), (50) меньше или равно bound (50), и (50) > 0 .
16	Напишите программу, которая вычисляет наибольший общий делитель двух чисел
17	Каждый день растение растет на метр вверх. Каждую ночь высота растения уменьшается на метры вниз из-за недостатка солнечного тепла. Первоначально высота растения равна 0 м. Мы сажаем семя в начале дня. Мы хотим знать, когда высота растения достигнет определенного уровня. **Пример** Для значений UpSpeed = 100, DownSpeed = 10 и desiredHeight = 910 выходное значение должно быть равно 10. После дня 1 --> 100 После ночи 1 --> 90

	После дня 2 --> 190 После ночи 2 --> 180 После дня 3 --> 280 После ночи 3 --> 270 После дня 4 --> 370 После ночи 4 --> 360 После дня 5 --> 460 После ночи 5 --> 450 После дня 6 --> 550 После ночи 6 --> 540 После дня 7 --> 640 После ночи 7 --> 630 После дня 8 --> 730 После ночи 8 --> 720 После дня 9 --> 820 После ночи 9 --> 810 После 10-го дня --> 910
18	Напишите функцию, которая осуществляет проверку на палиндром строки
19	Выполните функцию преобразования целого числа в строку с турецким именем числа. На вход всегда подается целое число 0-99; Выходные данные всегда должны быть в нижнем регистре. Формирование турецких названий для чисел 0-99 очень просто: • единицы (0-9) и десятки (10, 20, 30 и т.д.) имеют свое собственное уникальное имя; • все остальные числа обозначаются просто [tens] + [unit], как, например, twenty one в английском языке. В отличие от английского, в турецком языке нет чисел с суффиксом "teen"; например, 13 в английском языке переводится как "ten three", а не "thirteen". В турецком языке единицы и десятки называются следующим образом: 0 = sıfır 1 = bir 2 = iki 3 = üç 4 = dört 5 = beş 6 = altı 7 = yedi 8 = sekiz 9 = dokuz

	10 = on 20 = yirmi 30 = otuz 40 = kırk 50 = elli 60 = altmış 70 = yetmiş 80 = seksen 90 = doksan **Примеры** 1 --> "bir" 13 --> "on üç" 27 --> "yirmi yedi" 38 --> "otuz sekiz" 77 --> "yetmiş yedi" 94 --> "doksan dört"
20	Задана строка, состоящая из букв a, b и/или c, поменяйте местами буквы a и b (замените a на b и наоборот). При этом все случаи появления буквы c оставьте нетронутыми. **Пример**: 'acb' --> 'bca' 'aabacbaa' --> 'bbabcabb'

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases на Google Test.

Лабораторная работа № 2

Тема: Классы

Цель:

1. Закрепление навыков работы с классами
2. Закрепление навыков работы с динамической памятью на «куче»
3. Закрепление навыков работы с массивами

Порядок выполнения работы

- Ознакомиться с теоретическим материалом.
- Получить у преподавателя вариант задания.
- Реализовать задание своего варианта в соответствии с поставленными требованиями.
- Написать Unit-тесты с использованием Google Test.
- Создать репозиторий на GitHub.
- Отправить файлы лабораторной работы в репозиторий.
- 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

Используя в качестве образца класс **Array** (см. ниже), реализовать динамические контейнеры с использованием динамического массива.

- Каждый класс должен быть разделен на интерфейс и реализацию.
- Самостоятельно определить необходимые типы, поля и дополнительные методы.
- Реализовать генерацию исключений в конструкторах и методах при необходимости (использовать стандартные исключения).
- Реализовать арифметические операции: сложение, вычитание, копирование
- Реализовать операции сравнения: (больше, меньше, равно).
- Арифметические операции с присваиванием должны быть реализованы как методы класса.

```
class Array
{
public:
   Array();
   Array(const size_t & n, unsigned char t = 0);
   Array(const std::initializer_list< unsigned char> &t);
   Array(const string &t);
   Array(const Array& other);
   Array(Array&& other) noexcept;
   virtual ~Array() noexcept;
};
```

Классы:

1. Создать класс `Decimal` для работы с беззнаковыми целыми десятичными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является десятичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).

2. Создать класс `Hex` для работы с беззнаковыми целыми шестнадцатеричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является шестнадцатеричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
3. Создать класс `Octal` для работы с беззнаковыми целыми восьмеричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является восьмеричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
4. Создать класс `Four` для работы с беззнаковыми целыми чертвертичными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является четвертичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
5. Создать класс `Three` для работы с беззнаковыми целыми троичными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является троичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
6. Создать класс `Five` для работы с беззнаковыми целыми пятиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является пятеричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
7. Создать класс `Six` для работы с беззнаковыми целыми шестиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является шестиричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
8. Создать класс `Seven` для работы с беззнаковыми целыми семеричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является семиричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
9. Создать класс `Eleven` для работы с беззнаковыми целыми одиннадцатиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является одиннадцатиричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
10. Создать класс `Twelve` для работы с беззнаковыми целыми двенадцатиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является двенадцатиричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
11. Создать класс `Thirteen` для работы с беззнаковыми целыми тринадцатиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является тринадцатиричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
12. Создать класс `Money` для работы с денежными суммами. Сумма должна быть представлена массивом из элементов типа `unsigned char`, каждый элемент которого – десятичная цифра. Младший индекс соответствует младшей цифре денежной суммы. Младшие две цифры — копейки.
13. Создать класс `Binary` для работы с двоичными беззнаковыми числами фиксированной длины. Число должно быть представлено массивом типа `unsigned char`, каждый элемент которого принимает значение 0 или 1. Младший бит имеет младший индекс.
14. Создать класс `BitString` для работы с битовыми строками. Битовая строка должна быть представлена массивом типа `unsigned char`, каждый элемент которого принимает значение 0 или 1. Должны быть реализованы все традиционные операции для работы с битовыми строками: `and`, `or`, `xor`, `not`.

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases на Google Test.

Лабораторная работа № 03

Тема: Наследование, полиморфизм

Цель:

- Изучение механизмов работы с наследованием в C++;
- Изучение механизма перегрузки операций

Порядок выполнения работы

- Ознакомиться с теоретическим материалом.
- Получить у преподавателя вариант задания.
- Реализовать задание своего варианта в соответствии с поставленными требованиями.
- Написать Unit-тесты с использованием Google Test.
- Создать репозиторий на GitHub.
- Отправить файлы лабораторной работы в репозиторий.
- 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 (std::cin) и выводить данные в стандартный вывод (std::cout).

Необходимо зарегистрироваться на GitHub и создать репозитарий для задания лабораторных работ.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github. Необходимо реализовать функцию согласно варианту задания. Функция должна быть помещена в отдельный файл.

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения.

Все классы должны поддерживать набор общих методов:

1. Вычисление геометрического центра фигуры вращения;
2. Вывод в стандартный поток вывода std::cout координат вершин фигуры через перегрузку оператора << для std::ostream;
3. Чтение из стандартного потока данных фигур через перегрузку оператора >> для std::istream
4. Вычисление площади фигуры через перегрузку оператора приведения к типу double;

Создать программу, которая позволяет:

- Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
- Сохранять созданные фигуры в динамический массив (по аналогии с предыдущей лабораторной работой Array) указатели на фигуру (Figure*)
- Фигуры должны иметь переопределенные операции копирования (=), перемещения (=) и сравнения (==)

- Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать для каждой фигуры в массиве геометрический центр и площадь.
- Необходимо уметь вычислять общую площадь фигур в массиве.
- Удалять из массива фигуру по индексу;

Варианты заданий (выпуклые равносторонние фигуры вращения):

Вариант	Фигура №1	Фигура №2	Фигура №3
1.	Треугольник	Квадрат	Прямоугольник
2.	Квадрат	Прямоугольник	Трапеция
3.	Прямоугольник	Трапеция	Ромб
4.	Трапеция	Ромб	5-угольник
5.	Ромб	5-угольник	6-угольник
6.	5-угольник	6-угольник	8-угольник
7.	6-угольник	8-угольник	Треугольник
8.	8-угольник	Треугольник	Квадрат
9.	Треугольник	Квадрат	Прямоугольник
10.	Квадрат	Прямоугольник	Трапеция
11.	Прямоугольник	Трапеция	Ромб
12.	Трапеция	Ромб	5-угольник
13.	Ромб	5-угольник	6-угольник
14.	5-угольник	6-угольник	8-угольник
15.	6-угольник	8-угольник	Треугольник
16.	8-угольник	Треугольник	Квадрат
17.	Треугольник	Квадрат	Прямоугольник
18.	Квадрат	Прямоугольник	Трапеция
19.	Прямоугольник	Трапеция	Ромб
20.	Трапеция	Ромб	5-угольник
21.	Ромб	5-угольник	6-угольник
22.	5-угольник	6-угольник	8-угольник
23.	6-угольник	8-угольник	Треугольник
24.	8-угольник	Треугольник	Квадрат
25.	Треугольник	Квадрат	Прямоугольник
26.	Квадрат	Прямоугольник	Трапеция
27.	Прямоугольник	Трапеция	Ромб
28.	Трапеция	Ромб	5-угольник
29.	Ромб	5-угольник	6-угольник
30.	5-угольник	6-угольник	8-угольник
31.	6-угольник	8-угольник	Треугольник
32.	8-угольник	Треугольник	Квадрат

33.	Треугольник	Квадрат	Прямоугольник
34.	Квадрат	Прямоугольник	Трапеция
35.	Прямоугольник	Трапеция	Ромб
36.	Трапеция	Ромб	5-угольник

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases на Google Test.

Лабораторная работа № 04

Тема: Основы метапрограммирования

Цель:

- Изучение основ работы с шаблонами (template) в C++;
- Изучение шаблонов умных указателей

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 шаблоны классов согласно варианту задания. Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат. Фигуры являются фигурами вращения (равнобедренными), за исключением трапеции и прямоугольника (эти фигуры просто должны быть вписаны в круг).Для хранения координат фигур необходимо использовать шаблон std::pair (или реализовать свой шаблон template <class T> Point,в качестве параметра шаблона должен быть тип для переменных координат)

Например:

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

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

1. Вычисление геометрического центра фигуры вращения;
2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;
3. Вычисление площади фигуры;

Создать программу, которая позволяет:

- Запрещается использовать сырые указатели
- Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
- Сохранять созданные фигуры в динамический массив (переиспользовать от предыдущей лабораторной работы) умных указатели на фигуру (**std::shared_ptr<Figure[]>**)

- Динамический массив должен быть сделан в виде шаблона (параметр шаблона – класс для хранения в массиве `template <class T> Array {…})`
- Фигуры должны иметь переопределенные операции копирования, сравнения и приведение к типу double (вычисление площади)
- Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.
- Необходимо уметь вычислять общую площадь фигур в массиве.
- Удалять из массива фигуру по индексу;

Варианты заданий:

Вариант	Фигура №1	Фигура №2	Фигура №3
1.	Треугольник	Квадрат	Прямоугольник
2.	Квадрат	Прямоугольник	Трапеция
3.	Прямоугольник	Трапеция	Ромб
4.	Трапеция	Ромб	5-угольник
5.	Ромб	5-угольник	6-угольник
6.	5-угольник	6-угольник	8-угольник
7.	6-угольник	8-угольник	Треугольник
8.	8-угольник	Треугольник	Квадрат
9.	Треугольник	Квадрат	Прямоугольник
10.	Квадрат	Прямоугольник	Трапеция
11.	Прямоугольник	Трапеция	Ромб
12.	Трапеция	Ромб	5-угольник
13.	Ромб	5-угольник	6-угольник
14.	5-угольник	6-угольник	8-угольник
15.	6-угольник	8-угольник	Треугольник
16.	8-угольник	Треугольник	Квадрат
17.	Треугольник	Квадрат	Прямоугольник
18.	Квадрат	Прямоугольник	Трапеция
19.	Прямоугольник	Трапеция	Ромб
20.	Трапеция	Ромб	5-угольник
21.	Ромб	5-угольник	6-угольник
22.	5-угольник	6-угольник	8-угольник
23.	6-угольник	8-угольник	Треугольник
24.	8-угольник	Треугольник	Квадрат
25.	Треугольник	Квадрат	Прямоугольник
26.	Квадрат	Прямоугольник	Трапеция
27.	Прямоугольник	Трапеция	Ромб
28.	Трапеция	Ромб	5-угольник

29.	Ромб	5-угольник	6-угольник
30.	5-угольник	6-угольник	8-угольник
31.	6-угольник	8-угольник	Треугольник
32.	8-угольник	Треугольник	Квадрат
33.	Треугольник	Квадрат	Прямоугольник
34.	Квадрат	Прямоугольник	Трапеция
35.	Прямоугольник	Трапеция	Ромб
36.	Трапеция	Ромб	5-угольник

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases на Google Test.

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

1. Аллокатор
    a. Реализовать свой аллокатор памяти. Проверить что он корректно работает для контейнера std::map.
    b. Аллокатор должен параметризоваться количеством выделяемых за раз элементов.
    c. Освобождение конкретного элемента не предполагается - аллокатор должен освобождать всю память самостоятельно.
2. Контейнер
    a. Реализовать свой контейнер (согласно варианта задания), который по аналогии с контейнерами stl, параметрезуя аллокатором.
3. Итератор
    a. Реализовать итераторы (обычный и const)
    b. Итератор должен соответствовать std::forward_iterator_tag

Прикладной код должен содержать следующие вызовы:

- создание экземпляра std::map с созданным аллокатором
- заполнение 10 элементами, где ключ — это число от 0 до 9, а значение - факториал ключа
- вывод на экран всех значений (ключ и значение разделены пробелом) хранящихся в контейнере
- создание экземпляра своего контейнера для хранения int с собственным vаллокатором – заполнение контейнера и печать его элементов

Варианты заданий:

Вариант	Контейнер	Хранилище внутри аллокатора

1.	Динамический массив	std::vector
2.	Стек	std::vector
3.	Однонаправленный список	std::vector
4.	Двунаправленный список	std::vector
5.	Очередь	std::vector
6.	Динамический массив	std::deque
7.	Стек	std::deque
8.	Однонаправленный список	std::deque
9.	Двунаправленный список	std::deque
10.	Очередь	std::deque
11.	Динамический массив	std::list
12.	Стек	std::list
13.	Однонаправленный список	std::list
14.	Двунаправленный список	std::list
15.	Очередь	std::list
16.	Динамический массив	std::array
17.	Стек	std::array
18.	Однонаправленный список	std::array
19.	Двунаправленный список	std::array
20.	Очередь	std::array

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases на Google Test.

Лабораторная работа № 06

Тема: Паттерны проектирования

Цель:

- Закрепление паттернов SOLID

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

Вы разрабатываете редактор подземелья для новой RPG игры Balagur Fate 3. Ваша задача сделать программу, которая позволит моделировать базовые локации, расставляя на них NPC (non-player characters).

Редактор должен позволять делать следующие действия:

1. Добавлять новые NPC по разным координатам (типы объектов предопределены заданием)
    a. NPC размещаются в квадрате 0<=x<=500 метров, 0<=y<=500 метров
    b. У каждого типа объекта помимо координат должно быть уникальное имя.
2. Сохранять объекты в файл и загружать из файла
3. Печатать перечень объектов на экран (с типом объектов, координатами и именем)
4. Запускать боевой режим с указанием дальности, на которой могут сражаться NPC. В боевом режиме NPC начинают сражаться каждый с каждым (если дальность позволяет), проигравший NPC – удаляется. При этом погибнуть могут и оба NPC.

При создании программы обязательно использовать паттерны:

- Какой-нибудь из вариантов **Factory** для создания NPC и загрузки NPC из файла
- **Visitor** – для проведения цикла сражения
- **Observer** – для печати событий о совершенных убийствах. Сделать два класса Observer – для записи в файл “log.txt” и для печати на экран.

Варианты заданий:

Вариант	Объекты	Правило совместимости
1.	Орк, Белка, Друид	Орки убивают друидов Друиды убивают Белок Белки за мир

2.	Странствующий рыцарь, Эльф, Дракон	Дракон убивает всех (включая других драконов) Странствующий рыцарь убивает драконов Эльф убивает странствующих рыцарей (мстит за драконов)
3.	Орк, Белка, Медведь	Орки убивают медведей и орков Медведи убивают белок Белки не хотят воевать
4.	Белка, Эльф, Разбойник	Эльфы убивают разбойников Разбойники убивают белок Белки решили убивать эльфов (способ не известен)
5.	Медведь, Оборотень, Разбойник	Оборотни убивают Разбойников Разбойники убивают медведей Медведи убивают оборотней
6.	Медведь, Эльф, Разбойник	Эльфы убивают разбойников Разбойники убивают разбойников Медведь убивает Эльфов
7.	Орк, Странствующий рыцарь, Медведь	Орки убивают медведей Медведи убивают рыцарей Рыцари убивают Орков
8.	Белка, Оборотень, Друид	Белка убивает оборотня (не любит их) и друида (на всякий случай) Оборотень убивает друида Друид ни кого не обижает
9.	Эльф, Дракон, Друид	Дракон нападает на эльфов Эльф нападает на друида Друид нападает на дракона
10.	Разбойник, Орк, Оборотень	Разбойник убивает оборотней Оборотень убивает разбойника Орк убивает разбойника
11.	Странствующий рыцарь, Друид, Эльф	Рыцарь убивает эльфа Эльф убивает друида и рыцаря Друид убивает друидов
12.	Принцесса, Дракон, Странствующий рыцарь	Принцесса никого не трогает Дракон есть принцесс Рыцарь ест драконов

13.	Дракон, Жаба, Странствующий рыцарь	Жаба ест всех (включая жаб) Рыцарь убивает драконов Дракон убивает рыцарей
14.	Работорговец, Белка, Странствующий рыцарь	Работорговец убивает белку Рыцарь убивает работорговцев Белка убивает белок (наверное делят территорию)
15.	Странствующий рыцарь, Пегас , Дракон	Пегас никого не трогает Дракон есть пегаса Рыцарь убивает дракона
16.	Работорговец, Друид, Орк	Друид никого не трогает Орк убивает всех (включая орков) Работорговец убивает друидов
17.	Разбойник, Странствующий рыцарь, Эльф	Разбойник убивает эльфов Эльф убивает рыцарей Рыцарь убивает разбойников
18.	Странствующий рыцарь, Белка, Пегас	Рыцарь убивает белок Белки убивают пегасов Пегасы никого не трогают
19.	Медведь, Выпь, Выхухоль	Медведь ест всех кроме медведей Выпь никого не обижает Выхухоль убивает медведей (потому что она наш эндемик, а медведи они всякие бывают)
20.	Дракон, Бык, Жаба	Дракон ест быков Бык топчет жаб Жабы спасаются как могут

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.
3. Набор testcases на Google Test.

Лабораторная работа № 07

Тема: Асинхронное программирование

Цель:

- Знакомство с асинхронным программированием;
- Получение практических навыков в параллельной обработке данных;
- Получение практических навыков в синхронизации потоков;

Порядок выполнения работы

- Ознакомиться с теоретическим материалом.
- Получить у преподавателя вариант задания.
- Реализовать задание своего варианта в соответствии с поставленными требованиями.
- Подготовить тестовые наборы данных.
- Создать репозиторий на GitHub.
- Отправить файлы лабораторной работы в репозиторий.
- 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

Модифицируйте вашу лабораторную работу №6 следующим образом:

- Должно быть реализовано три потока:
    - Поток, который осуществляет передвижение NPC на определенное расстояние (см. таблицу), а также определяет, что два NPC находятся на расстоянии убийства (см. таблицу). Если два NPC вступили в «бой», то он создает задачу для потока, осуществляющего бои.
    - Поток, который осуществляет бои. Если один npc может убивать другой, то каждый npc «кидает 6-гранный кубик» определяя силу атаки и силу защиты (соответственно). Если сила атаки больше, чем сила защиты – то происходит убийство.
    - Мертвые npc не передвигаются (у нас тут без некромантов).
    - Живые npc не могут покинуть карту (размер задается через константы, например 100 x 100)
    - Основной поток раз в секунду печатает карту. Мертвые npc на карте не отображаются.
- Осуществить контроль доступа к разделяемым ресурсов с помощью std::shared_lock и std::lock_guard
- Осуществить контроль к потоку вывода std::cout через std::lock_guard
- Потоки могут запускаться как std::thread с использованием лямбда функций или функторов (классов с перегруженным оператором operator())
- Вначале игры должно создаваться 50 npc в случайных локациях.
- Игра должна останавливаться через 30 секунд и выводить на экран список выживших

Таблица убиваемости:

Кто нагоняет	Расстояние хода	Расстояние убийства

Орк	20	10
Белка	5	5
Друид	10	10
Странствующий рыцарь	30	10
Эльф	10	50
Дракон	50	30
Медведь	5	10
Разбойник	10	10
Оборотень	40	5
Принцесса	1	1
Жаба	1	10
Работорговец	10	10
Пегас	30	10
Выпь	50	10
Выхухоль	5	20
Бык	30	10

Отчет

1. Код программы на языке C++.
2. Ссылка на репозиторий на GitHub.