# Лабораторная работа № 04

# Тема: Основы метапрограммирования

**Цель:**

- Изучение основ работы с шаблонами (template) в C++;
- Изучение шаблонов std::pair, std::tuple
- Получение навыка работы со специализацией шаблонов и идиомой SFINAE

## 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_04** (в случае использования Windows **oop_exercise_04.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_04

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
- test_xx.txt // файл с тестовыми данными. Где xx – номер тестового набора 01, 02 , … Тестовых наборов должно быть больше 1.
- 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 только публичные поля. В классах не должно быть методов, только поля. Фигуры являются фигурами вращения (равнобедренными), за исключением 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;
};
```

Необходимо реализовать две шаблонных функции:

1. Функция **print** печати фигур на экран std::cout (печататься должны координаты вершин фигур). Функция должна принимать на вход std::tuple с фигурами, согласно варианту задания (минимум по одной каждого класса).
2. Функция **square** вычисления суммарной площади фигур. Функция должна принимать на вход std::tuple с фигурами, согласно варианту задания (минимум по одной каждого класса).

Создать программу, которая позволяет:

- Создает набор фигур согласно варианту задания (как минимум по одной фигуре каждого типа с координатами типа int и координатами типа double).
- Сохраняет фигуры в std::tuple
- Печатает на экран содержимое std::tuple с помощью шаблонной функции print.
- Вычисляет суммарную площадь фигур в std::tuple и выводит значение на экран.

При реализации шаблонных функций допускается использование вспомогательных шаблонов std::enable_if, std::tuple_size, std::is_same.

**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.
4. Результаты выполнения тестов.
5. Объяснение результатов работы программы.