Вариант 6

# Тема

Изучение свойств контейнеров STL переопределение операций. Математическое множество на основе set. Применение умных указателей weak\_ptr и shared\_ptr.

# Ответы на вопросы:

1. Какой операции вы придадите смысл объединения?

Операция объединения (operator+) для контейнера set.

1. Что будет происходить в результате применения операции? Изменятся ли исходные множества? Возникнет ли новое множество?

При использовании операции объединения (operator+) для множества set и unordered\_set:

Новое множество будет содержать все элементы из обоих исходных множеств a и b, без повторений, исходные множества a и b останутся неизменными, будет создано новое множество, содержащее объединение элементов множеств a и b.

1. Какого типа параметры будут у операции? По ссылке или по значению?

Параметры a и b передаются по значению, что означает, что функция создает копии исходных множеств для выполнения операции.

1. Можно ли создать копию объекта set?

Да. В программе (operator+) создает копию множества set и возвращает его.

1. Как добавить в set элементы другого set? Что будет, если добавляемый элемент уже существует?

Операции объединения (operator+) добавляют элементы другого set в исходное множество с помощью метода insert. Если добавляемый элемент уже существует в исходном множестве, он игнорируется.

1. Как проще всего пробежать по множеству?

В программе для этого используются оба способа: цикл for для unordered\_set и итератор для set.

1. Что должен возвращать оператор?

Операторы возвращают результаты объединения или пересечения множеств.

Ситуация, когда может быть полезно использовать weak\_ptr и понятие циклических ссылок:

Есть классы Parent и Child. Каждый объект класса Parent может иметь одного или несколько детей, но у каждого объект класса Child может быть только один Parent. Если мы используем shared\_ptr для представления связи между ними, может возникнуть циклическая зависимость. Parent ссылается на объект child, который ссылается обратно на тот же объект parent – это приводит к утечке памяти. Но если при связи parent и child использовать shared\_ptr, а при обратной связи - weak\_ptr, объект child получает доступ к объекту parent, но не увеличивает счетчик ссылок и, если объект parent уничтожается, weak\_ptr в объекте child становится недействительным, и тогда утчки памяти не будет.

Вывод: Использование weak\_ptr позволяет избежать утечек памяти в ситуациях, когда возможна циклическая зависимость между объектами.