

# Углублённое программирование на C++

Семинар #4

Возможности стандарта C++20/C++23

Кухтичев Антон



education

26 июня 2025 года

# Напоминание отметиться на портале

и оставить отзыв  
после лекции



# Содержание занятия

- Диапазоны (ranges) и виды (views)
- `std::expected`
- `std::flat_map`
- Задачи на чтение кода

# Диапазоны (ranges) и виды (views) (1)

- Диапазон (range) - группа элементов, по которой можно выполнять итерирование. Она предоставляет итератор `begin()` и маркер конца `end()`. Все контейнеры STL являются диапазонами.
- Вид (view) - это то, что вы применяете к диапазону и выполняете какую-то операцию.
- Символ `|` - синтаксический сахар для композиции функций. Вместо записи `C(R)` вы можете записать `R | C`.

## Диапазоны (ranges) и виды (views) (2)

```
int main() {  
    std::vector<int> myVec{-3, 5, 0, 7, 4};  
    std::sort(myVec.begin(), myVec.end());  
    for (auto v: myVec) std::cout << v << " ";  
}
```

```
int main() {  
    std::vector<int> myVec{-3, 5, 0, 7, 4};  
    std::ranges::sort(myVec);  
    for (auto v: myVec) std::cout << v << " ";  
}
```



# std::expected

```
template< class T, class E >  
class expected;
```

- Два состояния:
  - Ожидаемое значение (expected value) — результат успешной операции.
  - Непредвиденное значение (unexpected value) — информация об ошибке.
- has\_value() Проверяет наличие значения (true — успех).
- operator\*, operator-> Доступ к значению (только если has\_value() == true).
- value() Возвращает значение. Если ошибка — генерирует bad\_expected\_access.
- error() Возвращает ошибку (только если has\_value() == false).

## std::expected

```
enum class parse_error
{
    invalid_input,
    overflow
};

auto parse_number(std::string_view& str) -> std::expected<double, parse_error>
{
    const char* begin = str.data();
    char* end;
    double retval = std::strtod(begin, &end);

    if (begin == end)
        return std::unexpected(parse_error::invalid_input);
    else if (std::isinf(retval))
        return std::unexpected(parse_error::overflow);

    str.remove_prefix(end - begin);
    return retval;
}
```

# `std::flat_map`<sup>C++23</sup>





# flat\_map<sup>C++23</sup>

Это адаптер, который предоставляет функциональность ассоциативного контейнера, содержащего пары ключ-значение с уникальными ключами. Ключи сортируются с помощью функции сравнения Compare.

```
#include <flat_map>
```

```
template<  
    class Key,  
    class T,  
    class Compare = std::less<Key>,  
    class KeyContainer = std::vector<Key>,  
    class MappedContainer = std::vector<T>  
> class flat_map;
```

# flat\_map<sup>C++23</sup>

- Вставка и удаление медленнее, чем у `std::map`;
- Вставка и удаление делает существующие итераторы невалидными;

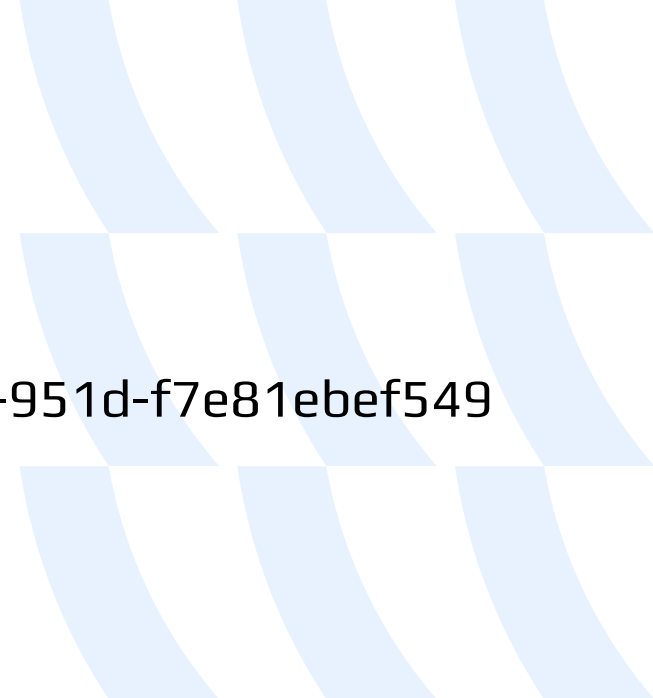
Вставка	Удаление	Поиск	Доступ
$O(n)$	$O(n)$	$O(\log n)$	-

# Задачи на чтение кода



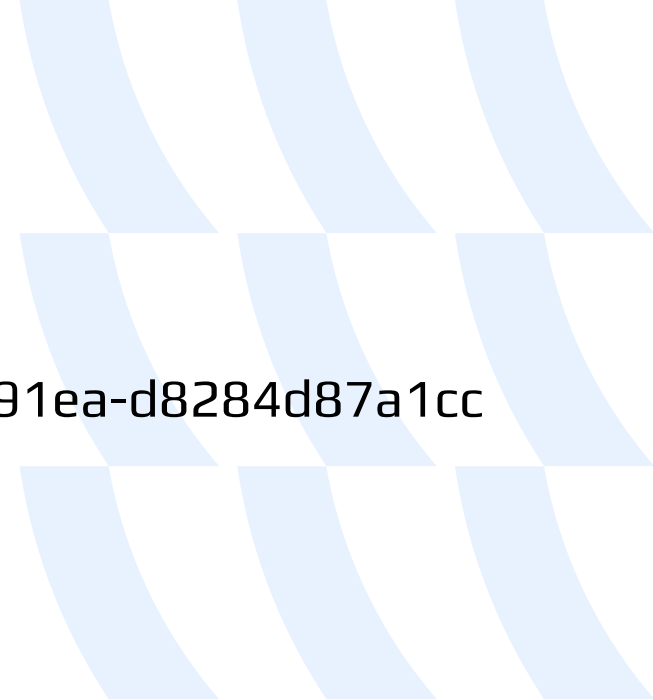
# Задача на чтение #1

<https://interview.cups.online/live-coding/?room=aeee886d-69e7-417b-951d-f7e81ebef549>



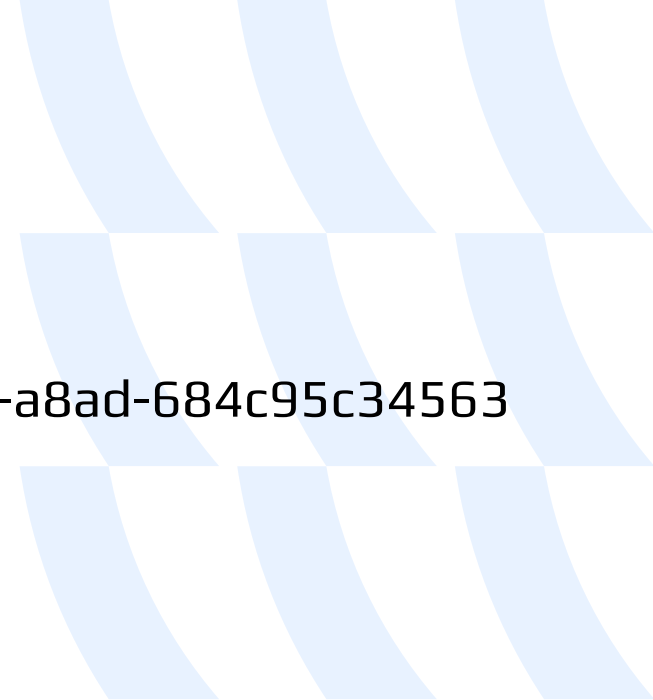
## Задача на чтение #2

<https://interview.cups.online/live-coding/?room=d2f53964-ec4f-4640-91ea-d8284d87a1cc>



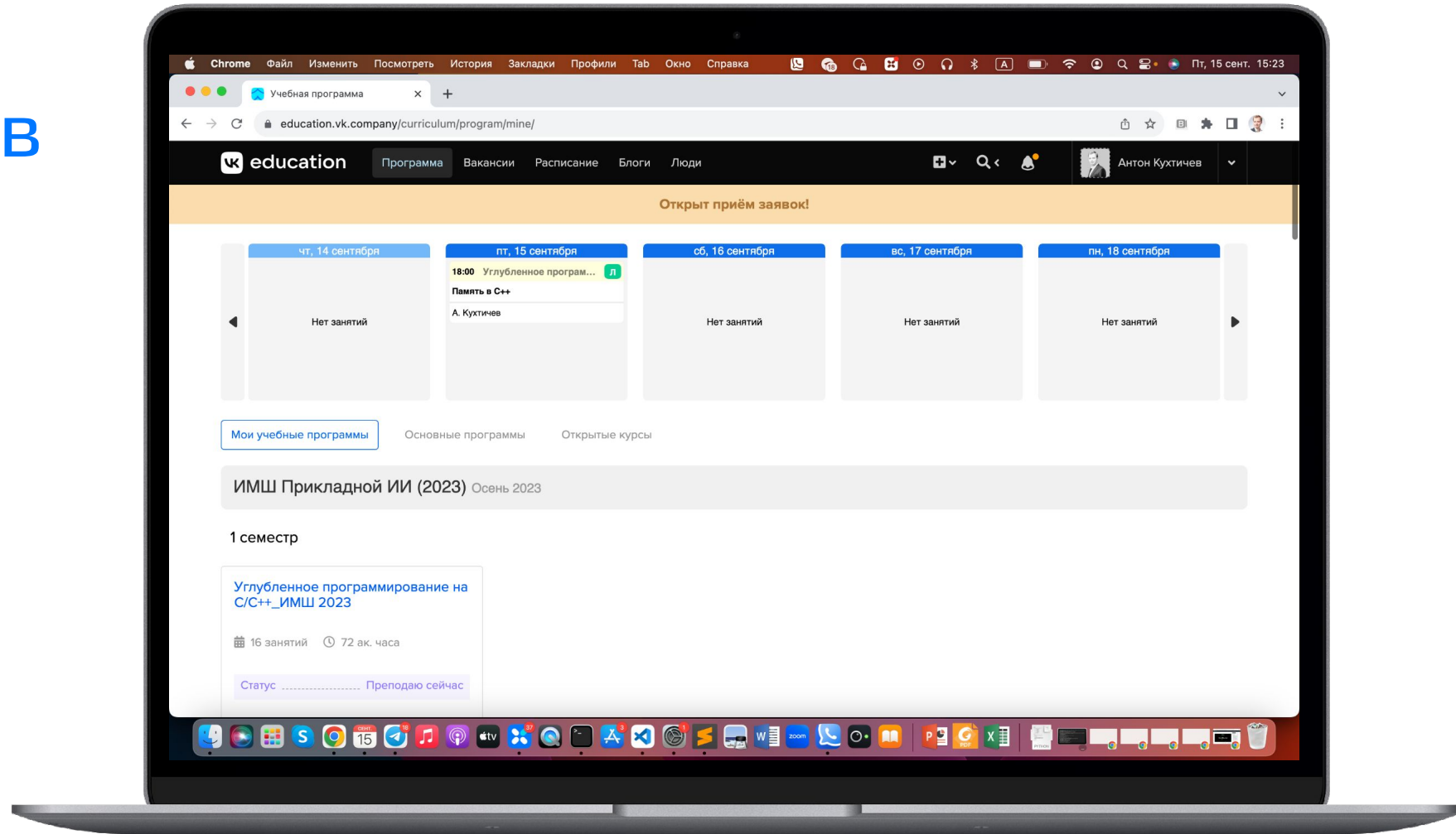
## Задача на чтение #3

<https://interview.cups.online/live-coding/?room=1119067c-769d-4d46-a8ad-684c95c34563>



# Напоминание оставить отзыв

Это правда важно





Спасибо  
за внимание!