ОНЛАЙН-ОБРАЗОВАНИЕ

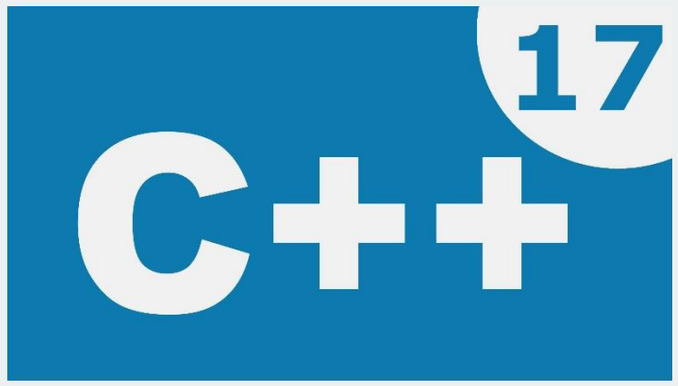
Разработчик C++

Идея аллокаторов

Сергей Кольцов
профессиональный программист

OTUS

Пятиминутка C++17



<cstddef>

std::byte – что это и зачем

<string_view>

std::string_view – полезный инструмент или источник ошибок?

.emplace_back с возвратом ссылки – мелочь, а приятно.

Что нас ждёт

- умные указатели и не только
- понятие аллокатора
- контейнеры и аллокаторы
- реализация аллокатора
- домашняя работа

Умные указатели

```cpp
auto ptr = std::make_unique<MyStruct>(42, "Vasia");

auto ptr2 = std::make_shared<MyStruct>(43, "Petia");
```

2 **Удобство создания – одна из приятных особенностей.**

Контейнеры

```
1. std::vector<int> values;
2. for(int i=0; i<10; ++i)
3.   values.emplace_back(i);
4.
5. some_func(values);
```

Аллокатор

```
1. template<
2.   class T,
3.   class Allocator = std::allocator<T>
4. > class vector;
5.
6. template<
7.   class Key,
8.   class T,
9.   class Compare = std::less<Key>,
10.  class Allocator = std::allocator<
11.       std::pair<const Key, T>
12.  >
13. > class map;
```

Управление памятью

- куча
- счётчик ссылок
- сборщик мусора

Куча так куча

```
1. void* malloc(std::size_t size);
2. void free(void* ptr);
3.
4. {
5.     void* p = malloc(sizeof(MyClass));
6.     MyClass* ptr = new(p) MyClass(42);
7.
8.     // ....
9.
10.    ptr->~MyClass();
11.    free(p);
12. }
```


Так лучше не делать никогда.

Как можно повлиять

- переопределить malloc
- переопределить operator new
- переопределить аллокатор для контейнера

Аллокатор

```
template<typename T>
struct logging_allocator {
using value_type = T;
// ....
```

Аллокатор

```
T *allocate(std::size_t n) {
        auto p = std::malloc(n * sizeof(T));
        if (!p)
            throw std::bad_alloc();
        return reinterpret_cast<T *>(p);
    }
```

Аллокатор

```
void deallocate(T *p, std::size_t n) {
        std::free(p);
    }
```

Аллокатор

```
template<typename U, typename ...Args>
void construct(U *p, Args &&...args) {
    new(p) U(std::forward<Args>(args)...);
};
```

Аллокатор

```
void destroy(T *p) {
        p->~T();
}
```

Тренировка

- реализовать свой аллокатор с поддержкой операции reserve
- использовать его в стандартном контейнере
- реализовать свой контейнер

Спасибо
за внимание!

Ответы на вопросы