

13.04.2024

Шаблоны классов

Спикер: Ибрагимов Булат Ленарович

Fast Track в Телеком, 2024

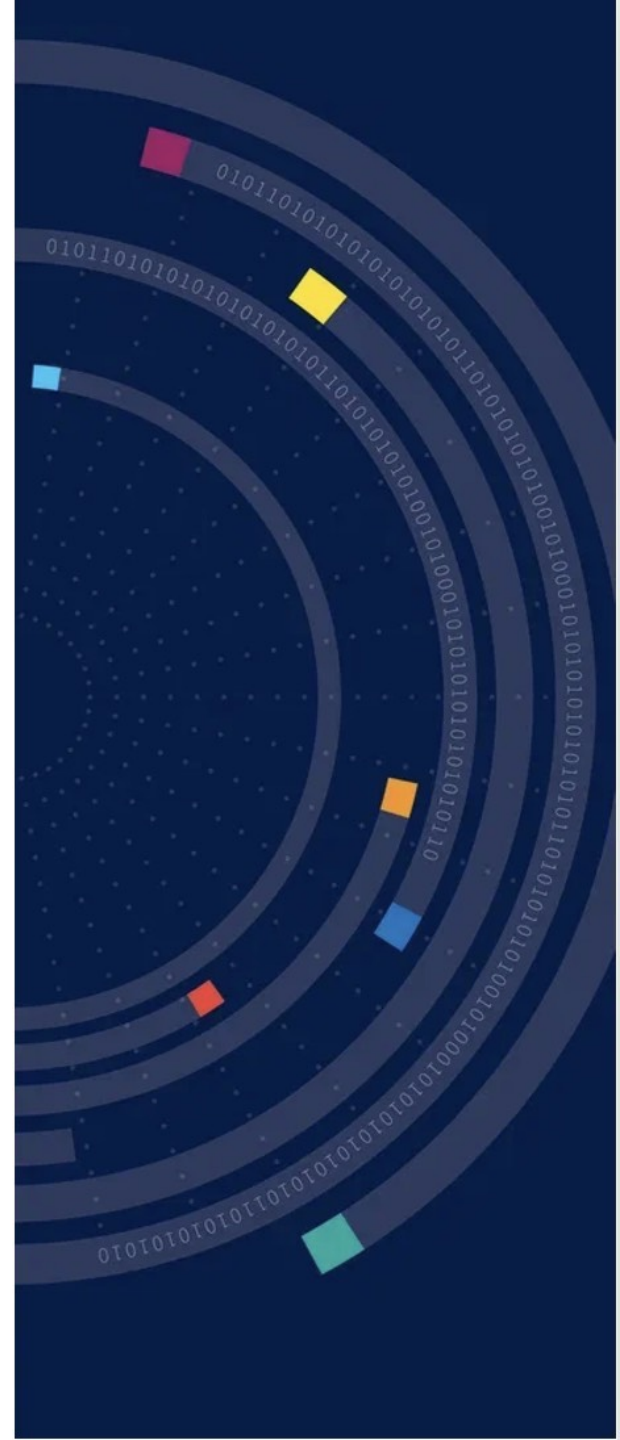




ИБРАГИМОВ БУЛАТ ЛЕНАРОВИЧ

- Преподаватель в МФТИ. Проводит курсы по С++ и алгоритмам, структурам данных
- Научный сотрудник Института Искусственного Интеллекта (AIRI)
- Работал разработчиком-исследователем в Яндекс и Сбербанк

Шаблоны классов



Мотивация

```
class StackInt {  
    public:  
        void Push(int value);  
        // ...  
};  
  
class StackDouble {  
    public:  
        void Push(double value);  
        // ...  
};  
  
class StackUnsigned {  
    public:  
        void Push(unsigned value);  
        // ...  
};
```

Синтаксис шаблонов классов

```
template <class T>
class Stack {
    inline static const size_t kMaxSize = 100;
    T buffer_[kMaxSize];
    size_t size_;

public:
    void Push(const T& value);
    void Pop();
    size_t Size() const;
    // ...
};

Stack<int> stack_int;           // [T=int]
Stack<double> stack_double;    // [T=double]

stack_int.Push(1);             // 1
stack_double.Push(1);          // 1.0

// Stack<int> и Stack<double> - абсолютно разные типы!
```

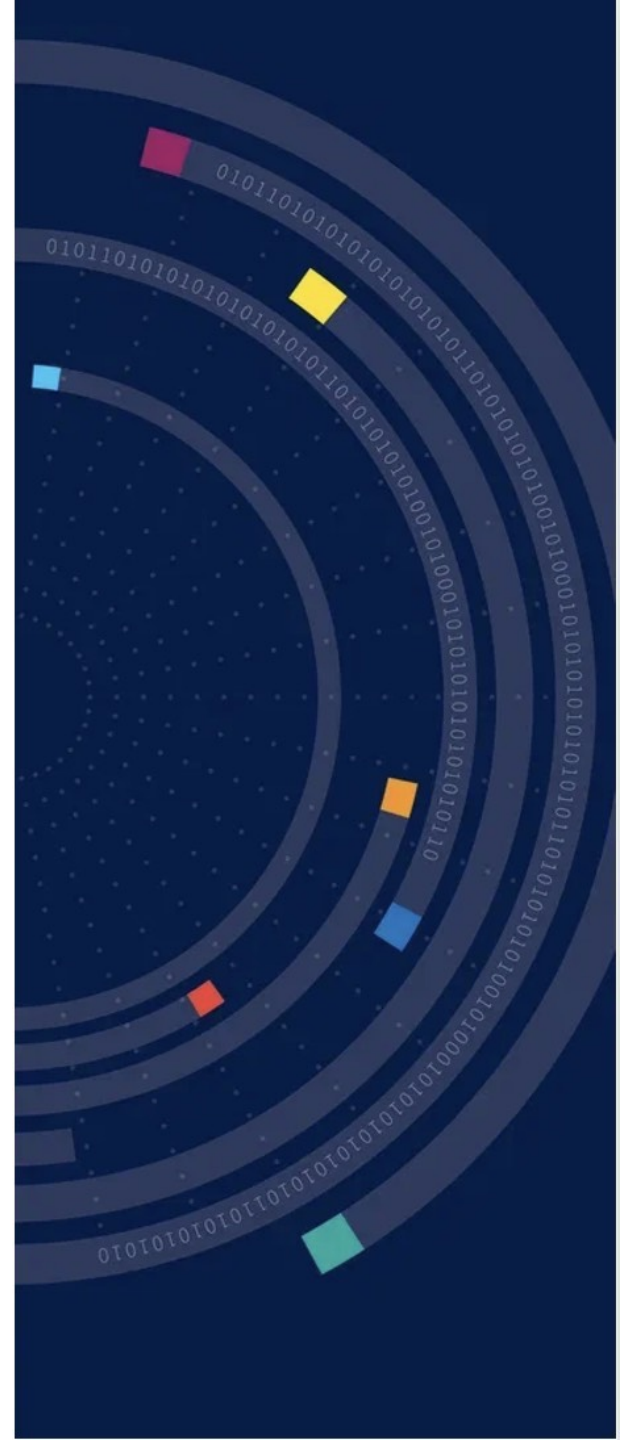
Шаблоны классов

- В отличие от обычных классов можно объявлять только в области видимости пространства имен, либо внутри другого класса.
- Тип шаблонного параметра нужно указывать явно (C++17: если невозможно вывести тип по конструктору).
- Шаблоны классов (как и другие шаблоны) инстанцируются "лениво". Более того, методы шаблонного класса тоже инстанцируются "лениво".

```
template <class T>
struct S {
    void f();    // инстанцируется
    void g();    // не инстанцируется
};

S<int> s;
s.f();
```

Специализация шаблонов



Специализация шаблонов классов

```
// общий шаблон
template <class T>
struct IsInt {
    inline static const bool value = false;
};

// полная специализация
template <>
struct IsInt<int> {
    inline static const bool value = true;
};
```

Как и в случае шаблонов функций шаблон и специализация могут кардинально отличаться (разные поля, разные методы).

```
template <>
struct IsInt<bool* const> {
    void Print() const { std::cout << "No"; }
}
```


Частичная специализация шаблонов классов

Иногда хочется задать определенное поведение класса не для конкретного типа, а для целого семейства типов (например, для указателей).

```
template <class T>    // общий шаблон
struct IsPointer {
    inline static const bool value = false;
    static bool IsIntPtr() { return false; }
};

template <class T>    // частичная специализация
struct IsPointer<T*> {
    inline static const bool value = true;
    static bool IsIntPtr() { return false; }
};

template <>           // полная специализация
struct IsPointer<int*> {
    inline static const bool value = true;
    static bool IsIntPtr() { return true; }
};
```

Частичная специализация шаблонов классов

Более специализированная версия всегда побеждает менее специализированную

```
std::cout << IsPointer<int>::value << ' '           // 0
          << IsPointer<int>::IsIntPtr() << '\n';    // 0

std::cout << IsPointer<char*>::value << ' '         // 1
          << IsPointer<char*>::IsIntPtr() << '\n';  // 0

std::cout << IsPointer<int*>::value << ' '          // 1
          << IsPointer<int*>::IsIntPtr() << '\n';  // 1
```

Частичная специализация шаблонов классов

```
template <class T, class U>  
struct S {  
};
```

```
template <class T>  
struct S<T, int> {  
};
```

```
template <class T>  
struct S<float, T> {  
};
```

```
S<bool, bool> a;  
S<bool, int> b;  
S<float, bool> c;  
S<float, int> d;
```

Частичная специализация шаблонов функций

- Ее не существует

Шаблонные параметры шаблонов

Шаблон может принимать другие шаблоны в качестве параметров

```
template <class T>
class Array {
// ...
};

// шаблон, принимающий тип и шаблон, принимающий тип
template <class T, template <class> class Container>
class Stack {
    Container<T> buffer;

    public:
// ...
};

Stack<int, Array> stack;
```