



# Cvičení C++

19.11.2018

[faltin@ksi.mff.cuni.cz](mailto:faltin@ksi.mff.cuni.cz)



# Šablony – základy (1/3)

```
template<typename T>
class complex {
    T real_; T im_;
public:
    T &re() { return real_; }
    T re() const { return real_; };
    T &im() { return im_; };
    T im() const { return im_; };
    // ...
};
```

# Šablony – základy (2/3)

```
template<class Cont>
class container_wrapper {
    Cont cont_;

public:
    void push_back(const typename Cont::value_type& val) {
        cont_.push_back(val);
    }

    void push_back(typename Cont::value_type &&rval) {
        cont_.push_back(std::move(rval));
    }

    // ...
};

template<class Cont>
container_holder<Cont> make_wrapper() {
    return container_holder<Cont>();
}
```




# Šablony – základy (3/3)

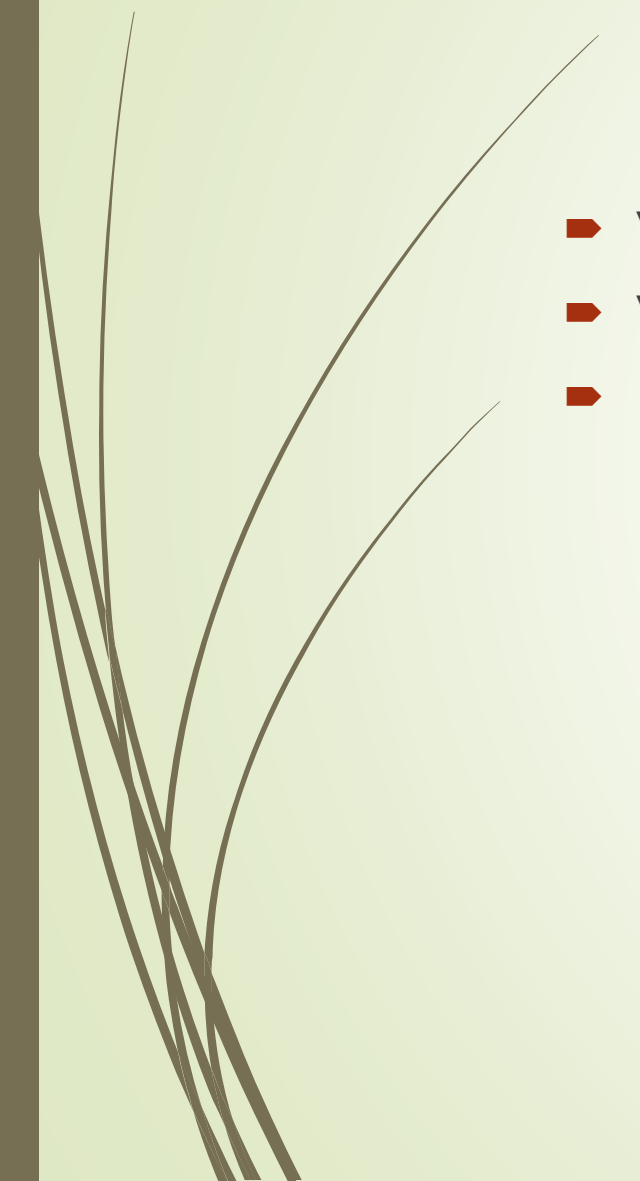
```
template<typename T, std::size_t Size>
class my_array {
    T data_[Size];
    // ...
};
```

```
template<typename T1, class T2, std::size_t N, double D, bool B>
struct S {};
```

```
S<std::string, std::vector<int>, 3, 3.2, false> s;
```



# Domácí úkol – „databáze“

- V ReCodexu - <https://recodex.mff.cuni.cz>
  - Variace na databázi
  - 14 dní
- 



# Úkoly



## 1. Vlastní implementace `std::vector<T>`

### 1. Pro libovolné objekty

1. `push_back()`, `size()`, `pop_back()`, `front()`, `back()`, `insert()`, `clear()`,
2. automaticky správně reallokuje

## 2. Vlastní implementace `std::array<T, size_t>`

## 3. Polymorfní vector

### 1. Pro libovolné objekty které dědí od nějakého interface `istorable`

### 2. Pro úplně libovolné objekty

- `poly_array.push_back(3).push_back("Ahoj").push_back(3.2);`
- `auto three = poly_array[0]; auto hello = poly_array[1]; auto three_two = poly_array[2];`