Программирование на языке C++
Лекция 10

Исключения в деструкторах и конструкторах

Александр Смаль

Исключения в деструкторах

Исключения не должны покидать деструкторы.

- Двойное исключение:

```
void foo() {
    try {
        Bad b; // исключение в деструкторе
        bar(); // исключение
    } catch (std::exception & e) {
        // ...
    }
}
```

- Неопределённое поведение:

```
void bar() {
    Bad * bad = new Bad[100];
    // ислючение в деструкторе №20
    delete [] bad;
}
```

Исключения в конструкторе

Исключения — это единственный способ прервать конструирование объекта и сообщить об ошибке.

```
struct Database {
    explicit Database(string const& uri) {
        if (!connect(uri))
            throw ConnectionError(uri);
    }
    ~Database() { disconnect(); }
    // ...
};
int main() {
    try {
        Database * db = new Database("db.local");
        db->dump("db-local-dump.sql");
        delete db;
    } catch (std::exception const& e) {
        std::cerr << e.what() << '\n';
    }
}
```

Исключения в списке инициализации

Позволяет отловить исключения при создании полей класса.

```
struct System
{
    System(string const& uri, string const& data)
    try : db_(uri), dh_(data)
    {
        // тело конструктора
    }
    catch (std::exception & e) {
        log("System constructor: ", e);
        throw;
    }

    Database    db_;
    DataHolder  dh_;
};
```