

C/C++: Лекция 8

Воробьев Д.В

23.10.2020

Исключения в деструкторах

```
struct X {
    std::string name;
    X(std::string name_) :
        name(std::move(name_)) {
        std::cout << "X" << name_;
    }
    ~X() noexcept(false) {
        std::cout << "~X:" << name;
        throw 1;
    }
};
```

```
void bar() {
    X a("a");
}

void foo() {
    X b("b");
    bar();
}

int main() {
    try {
        foo();
    } catch (...) {
        // handling
    }
    return 0;
}
```

std::uncaught_exception

Возвращает True \Leftrightarrow есть необработанное исключение

```
// до C++11  
bool uncaught_exception() throw();  
  
// с C++11  
bool uncaught_exception() noexcept;
```

std::uncaught_exception

```
struct Foo {
    ~Foo() {
        if(std::uncaught_exception()) {
            std::cout << "~Foo() : normall call";
        } else {
            std::cout << "~Foo() : stack unwinding";
        }
    }
};

int main() {
    Foo f;
    try {
        Foo f;
        throw std::runtime_error("test");
    } catch (const std::exception& e) {
        std::cout << e.what();
    }
}
```

Гарантии безопасности при исключениях

примеры взяты: CppCon 2019: Ben Saks
“Back to Basics: Exception Handling and Exception Safety”

No guarantee

```
class file {  
    public:  
        file(char const* name, char const* mode);  
        ~file() noexcept;  
        bool is_open() const noexcept;  
        void put(int i);  
        void put(char const* s);  
    private:  
        FILE* pf;  
};
```

No guarantee

```
file::file(char const* name, char const* mode) :  
    pf {fopen(name, mode)}  
{  
}  
  
file::~file() noexcept {  
    if (pf != nullptr) {  
        fclose(pf);  
    }  
}
```

No guarantee

```
void f(char const* n) {  
    FILE* outf = fopen(n, "w");  
    if (outf != nullptr) {  
        fprintf(outf, "The value is: %d", g());  
        fclose(outf);  
    }  
}
```


Basic guarantee

В функции породилось исключение \Rightarrow по завершении функции не будет утечки ресурсов

Basic guarantee

```
void f(char const* n) {  
    file outf(n, "w");  
    if (outf.is_open()) {  
        outf.put("The value is: ");  
        outf.put(g());  
    }  
}
```

Strong guarantee

В функции породилось исключение \Rightarrow функция не исполняется

"commit or rollback semantics" = "операция применяется полностью или не применяется вообще"

Strong guarantee

```
void f(char const* n) {  
    // Part 1  
    int temp = g();  
  
    // Part 2  
    file outf(n, "w");  
    if (outf.is_open()) {  
        outf.put("The value is:");  
        outf.put(temp);  
    }  
}
```

Nothrow guarantee

В функции породилось исключение \Rightarrow Исключение никогда не покинет функцию.

Nothrow guarantee

```
void f(char const* n) noexcept {  
    try {  
        file outf(n, "w");  
        if (outf.is_open()) {  
            outf.put("The value is: ");  
            outf.put(g());  
        }  
    } catch(...) {  
        //handling  
    }  
}
```