Программирование на языке C++
Лекция 10

Спецификация исключений

Александр Смаль

Спецификация исключений

Устаревшая возможность C++, позволяющая указать список исключений, которые могут быть выброшены из функции.

```
void foo() throw(std::logic_error) {
    if (...) throw std::logic_error();
    if (...) throw std::runtime_error();
}
```

Если сработает второй `if`, то программа аварийно завершится.

```
void foo() {
    try {
        if (...) throw std::logic_error();
        if (...) throw std::runtime_error();
    } catch (std::logic_error & e) {
        throw e;
    } catch (...) {
        terminate();
    }
}
```

Ключевое слово `noexcept`

- Используется в двух значениях:
  - Спецификатор функции, которая не бросает исключение.
  - Оператор, проверяющий во время компиляции, что выражение специфицированно как небросающее исключение.
- Если функцию со спецификацией `noexcept` покинет исключение, то стек не обязательно будет свёрнут, перед тем как программа завершится.
  В отличие от аналогичной ситуации с `throw()`.
- Использование спецификации `noexcept` позволяет компилятору лучше оптимизировать код, т.к. не нужно заботиться о сворачивании стека.

Использование noexcept

```
void no_throw() noexcept;
void may_throw();

// копирующий конструктор noexcept
struct NoThrow { int m[100] = {}; };

// копирующий конструктор noexcept(false)
struct MayThrow { std::vector<int> v; };
```

```
MayThrow mt;
NoThrow  nt;

bool a = noexcept(may_throw());  // false
bool b = noexcept(no_throw());   // true

bool c = noexcept(MayThrow(mt)); // false
bool d = noexcept(NoThrow(nt));  // true
```

Условный noexcept

В спецификации noexcept можно использовать условные выражения времени компиляции.

```
template <class T, size_t N>
void swap(T (&a)[N], T (&b)[N])
        noexcept(noexcept(swap(*a, *b)));

template <class T1, class T2>
struct pair {
    void swap(pair & p)
        noexcept(noexcept(swap(first, p.first)) &&
                 noexcept(swap(second, p.second)))
    {
        swap(first, p.first);
        swap(second, p.second);
    }

    T1 first;
    T2 second;
};
```

Зависимость от noexcept

Проверка `noexcept` используется в стандартной библиотеке для обеспечения строгой гарантии безопасности исключений с помощью `std::move_if_noexcept` (например, `vector::push_back`).

```
struct Bad {
    Bad() {}
    Bad(Bad&&);       // может бросить
    Bad(const Bad&); // не важно
};
struct Good {
    Good() {}
    Good(Good&&) noexcept; // не бросает
    Good(const Good&) ;    // не важно
};
```

```
Good g1;
Bad  b1;
Good g2 = std::move_if_noexcept(g1); // move
Bad  b2 = std::move_if_noexcept(b1); // copy
```