Проектирование больших систем на C++

Коноводов В. А.

кафедра математической кибернетики ВМК

Лекция 11
23.11.2018

noexcept

```
void f(int param) noexcept;
```

- Модификатор означает, что функция гарантированно не генерирует исключений.
- От этого может зависеть эффективность вызывающего кода.
- Допускается вызов из noexcept-функции других функций, которые noexcept не являются.
- В C++11 все функции освобождения памяти и все деструкторы неявно являются noexcept.

Гарантии безопасности исключений

1. Гарантия отсутствия исключений.
2. Строгая гарантия (исключения могут происходить, но все объекты остаются в согласованном и предсказуемом состоянии).
3. Базовая гарантия (исключения могут происходить, объекты остаются в согласованном состоянии, но не обязательно в предсказуемом).

Гарантии безопасности исключений

1. Гарантия отсутствия исключений.
2. Строгая гарантия (исключения могут происходить, но все объекты остаются в согласованном и предсказуемом состоянии).
3. Базовая гарантия (исключения могут происходить, объекты остаются в согласованном состоянии, но не обязательно в предсказуемом).

Пример: стек и операция `pop`.

- Согласованность означает соответствие size() и числа элементов при исключении.
- Предсказуемость означает, что при исключении число элементов в стеке не уменьшилось.

Assert

Проверяем предположения о данных объекта в программе, проверяем инварианты.

- assert времени компиляции

Программа не компилируется, если условие не выполняется.

```
static_assert(
    std::is_pointer<decltype(TCalcerPtr)>::value,
    "must be pointer"
);
```

Assert

- assert времени выполнения

```
#ifdef NDEBUG
#define assert(condition) ((void)0)
#else
#define assert(condition) /*implementation defined*/
#endif
```

А полезно ли это?

Readability, Correctness, Efficiency

- Хорошо ли написан код?
- Корректно ли написан код?
- Эффективно ли работает код?

- Тестирование
- Формальная верификация (автоматическая/полу-автоматическая)
- ...

Тестирование

- Цель тестирования кода?
- Как искать баги в коде и упростить себе жизнь?
  - Хороший дизайн кода и code review
  - unit-тестирование
  - Интеграционное тестирование
  - UI-тесты
  - Мониторинги
  - Acceptance тестирование

Тестирование

Unit-тесты (модульное тестирование):

- Быстрые
- Полные для своего компонента

Интеграционные тесты:

- Медленные
- Проверить всё невозможно
- Могут чаще флапать

Unit-тестирование

Задача: добавляем private-метод в класс с 100500 методами и полями. Нужно протестировать этот метод.

Behavior and State Based Testing

- Тесты, проверяющие что метод или функция отработали корректно, проверяют состояния объекта после вызова.
- Тесты на корректность самого взаимодействия объектов с другими объектами (а не результата).

gtest/gmock

Google C++ Testing Framework.

- Открытая библиотека для unit-тестирования.
- Ключевое понятие — assert
- Результат выполнения:
  - success
  - nonfatal failure
  - fatal failure
- Тест — набор assert'ов
- Набор тестов — тестовая программа или testing suite

gtest

```
ASSERT_TRUE(condition);
ASSERT_FALSE(condition);
ASSERT_EQ(val1, val2);
ASSERT_NE(val1, val2);
ASSERT_LT(val1, val2);
ASSERT_LE(val1, val2);
ASSERT_GT(val1, val2);
ASSERT_GE(val1, val2);
ASSERT_FLOAT_EQ(val1, val2);
ASSERT_DOUBLE_EQ(val1, val2);
ASSERT_NEAR(val1, val2, abs_error);

// strings
ASSERT_STREQ(str1, str2);
ASSERT_STRNE(str1, str2);
ASSERT_STRCASEEQ(str1, str2);
ASSERT_STRCASENE(str1, str2);
```

gtest

```
// exceptions
ASSERT_THROW(statement, exception_type);
ASSERT_ANY_THROW(statement);
ASSERT_NO_THROW(statement);
```

Можно добавлять message:

```
ASSERT_EQ(1, 2) << "i dont know why, but 1 != 2";
```

- `TEST` — макрос для определения теста
- `RUN_ALL_TESTS()` — запуск всех тестов

Тестирование и зависимости

- Зависимости классов
- Как обеспечить изоляцию?

Основные подходы:

- Dummy. Передаются в тестируемые классы/методы/функции в виде параметра без поведения, внутри как правило с ним ничего не происходит
- Stub. Подменяется внешняя зависимость, игнорируются все данные, входящие в stub из тестируемого объекта.
- Fake. Замена легковесной реализацией.
- Mock. Объекты, которые имитируют поведение реальных. Полезно, когда настоящие объекты непрактично/невозможно вставлять в юнит-тест, но поведение нужно сохранить.