

Шаблоны классов. Часть 2

1. Ключевое преимущество C++ — обобщенное программирование. Однако у данного механизма есть огромный минус — возникающие ошибки, диагностировать которые можно, но это:

- сложно
- не всегда возможно в принципе

Сформулируем *первую проблему* обобщенного программирования в C++: *ошибки при использовании шаблонов совершенно нечитаемы и диагностируются не там где сделаны, а в шаблоне.*

Вторая проблема: трудно писать разные реализации одной шаблонной функции для разных категорий типов. Обе проблемы легко решить, если добавить в язык всего одну возможность — накладывать ограничения на шаблонные параметры. Например, требовать, чтобы шаблонный параметр был контейнером или объектом. Это и есть концепт.

Концепты — это новая возможность, начиная с C++ 20. По сути это типизация аргументов шаблонов. Это позволяет предотвратить, или по крайней мере, уменьшить рискпередачи параметром шаблона произвольных объектов.

2. Объявление friend-функции в `Stack<>` будет считаться нешаблонным и возникнет ошибка времени исполнения (если компилятор не MSVC). Для исправления необходимо определить оператор «предварительно» вне класса. Перед определением объявить класс `Stack<>`. В самом классе объявить friend-функцию со специализацией (`<Type>`).
3. Специализация шаблона — его особая реализация для конкретного типа, т.к. иногда может потребоваться, чтобы один и тот же шаблонный класс вел себя одинаково для одних типов и по-другому для конкретного.

Например, обобщенный класс `Stack`

```
template<typename T>
class Stack
{
    ....
};
```

Специализация для `Matrix`

```
template<>
class Stack<Matrix>
{
    ....
};
```

4. Частичная специализация — реализация обобщенного класса для отдельного «семейства» типов, например, для указателей, потому что указатели так же как и обычные переменные могут иметь целочисленный, вещественный и т.п. Например, обобщенный класс `Stack`

```
template<typename T>
class Stack
{
    ....
};
```

Частичная специализация для указателей

```
template<typename T*>
class Stack
{
    ....
};
```