Шаблоны классов. Часть 2.

Основная идея концепта состоит в том, что концепты позволяют выявлять логические несоответствия между свойствами определённых типов

Они используются для обозначения множ-ва ограничений.

Т.к оператор << должен быть реализован как свободная функция, то оператор << для Stack<> не является шаблоном функции, а является обычной функцией инстанцируемой при необходимости с использованием шаблона класса.

Шаблонная реализация оператора <<
остаётся дружней
но теперь в классе Stack добавим
дружескую специализацию данного
шаблона

```
template <typename T>
class Matrix;

template <typename T>
std::ostream& operator<< (std::ostream &os,
                          const Matrix<T> &rhs)
{
   ... //code
}

template <typename T>
class Matrix
{
private
   T data;
   friend ostream& operator<< <T>(
                   ostream&, const Matrix&);
```

Что такое специализация шаблона класса

Как и в случае специализации шаблонов функций, можно специализировать шаблоны классов, оптимизируя и переопределяя поведение некоторых методов

```
template <>
class Stack<std::string> {
  ...
};
```

Частичная специализация шаблона класса

Мы можем определить частные реализации для определённых условий, при этом некоторые параметры шаблона всё ещё остаются задаваемые пользователем

Например: можно определить отдельную специализацию класса Stack для указателей