Задачи

Решение задач можно присылать (предпочтительнее ссылка на github, и т. п.) на `vkonovodov@gmail.com`. Бонусные баллы за решения задач могут быть поставлены нескольким первым приславшим правильное решение. Решение задач должно быть оформлено в виде компилирующегося кода.

(1) Стандарт C++11 добавляет средства для создания т. н. пользовательских литералов (user-defined literals) для суффиксов. Определите структуру для строк с кастомными литералами

```
template <char ...c> struct TString { };
```

и необходимые операторы так, чтобы компилировался код:

```
constexpr auto hello = "hello"_s + " world"_s;
static_assert(hello == "hello world"_s);
```

(2) Определите класс `TMyException`, унаследовав его от одного из стандартных исключений, определите в нем оператор `<<` для записи в объект исключения сопровождающего текста при его генерации. Породите из него несколько классов-наследников. Продемонстрируйте перехват исключений этого типа с помощью теста `EXPECT_THROW`.

(3) Реализуйте указатель `TIntrusivePtr`, создав базовый класс `TBasePtr`, в котором определите операторы `->`, `*`, `==`, `!=`, `bool` (предполагая, что отнаследовав от этого класса, можно реализовать и другие умные указатели). В основном классе должны быть определены основные копирующие и перемещающие операции, методы `UseCount`, `Get`, `Reset`, `Release`, а также конструктор от обычного указателя.

Кроме того, определите функцию `MakeIntrusive`, создающую этот указатель по аналогии с функцией `make_shared`.

Указатель `TIntrusivePtr` использует механизм встроенного подсчета ссылок. Счетчик ссылок находится не в самом указателе, как в `std::shared_ptr`, а в объекте. Поэтому необходимо определить также класс `TRefCounter`, представляющий собой простейший счетчик ссылок. У него должен быть определен, в частности, метод `RefCount`, возвращающий число указателей `TIntrusivePtr`, связанных с данным объектом.

Использование должно быть таким:

```
#include "intrusive_ptr.h"
class TDoc: public TRefCounter<TDoc> {
    // ...
}

// ...

TIntrusivePtr<TDoc> ptr = nullptr;
ptr = MakeIntrusive<TDoc>();     // ptr.RefCount() == 1
TIntrusivePtr<TDoc> ptr2 = ptr; // ptr.RefCount() == 2
TIntrusivePtr<TDoc> ptr3 = MakeIntrusive<TDoc>();
ptr3.Reset(ptr2);                // ptr.RefCount() == ptr3.RefCount() == 3
ptr3.Reset();                    // ptr.RefCount() == ptr3.RefCount() == 2
ptr3.Reset(std::move(ptr2));     // ptr.RefCount() == ptr3.RefCount() == 2
static_assert(sizeof(TDoc*) == sizeof(ptr));
```

Указанный выше код должен быть рабочим.