

3.30 Мотивировка move-семантики. Какую проблему она решает? Пример ситуации, когда уместно использовать `std::move`. Пример правильной реализации move-конструктора и move-оператора присваивания для класса `String`.

Мотивировка move-семантики:

1. Когда мы хотим передать объект таким образом, мы вынуждены идти на одно "одноразовое" копирование

```
1 int main() {
2     vector<string> v;
3     v.push_back(string("abc"));
4 }
```

2. Когда мы делаем реаллокацию вектора, старые объекты нужно переложить на новое место. Придётся проделывать полное копирование.
3. Рассмотрим реализацию функции `swap`.

```
1 template <typename T>
2 void swap(T& x, T& y) {
3     T tmp = x;
4     x = y;
5     y = tmp;
6 }
```

В данном коде наблюдается тройное копирование в случае сложных объектов.

```
41 template <typename T>
42 T create(...) {
43     //...
44     return obj;
45 }
46
47 int main() {
48     vector<string> v;
49     v.push_back(string("abc"));
50     //получаем лишнее копирование
51     foo(create());
52 }
```

Пример использования:

```
1 template <typename T>
2 void swap(T& x, T& y) {
3     T tmp = std::move(x);
4     x = std::move(y);
5     y = std::move(tmp);
6 }
```

После того, как объект мувают, все его параметры возвращаются к дефолтным, например, размер мувнутой строки равен нулю. **К объекту, который мувнули, можно обращаться, гарантируется, что он останется в валидном состоянии.**

Если же объект мувнули и результат действия нигде не используется, то объект остаётся нетронутым:

```

1 int main() {
2     string s("abc");
3     std::move(s);
4 }

```

Пример правильной реализации move-конструктора и move-оператора присваивания для класса String

```

1 String(String&& s): sz(s.sz), s(s.str) {
2     s.str = nullptr;
3     s.sz = 0;
4 }
5
6 String& operator=(String&& s) {
7     String temp = std::move(s);
8     swap(temp);
9     return *this;
10 }

```

3.31 Мотивировка умных указателей. Какую проблему они решают? Базовый синтаксис использования shared_ptr (как создать, как пользоваться в простейшем случае)

см. билет 4.28.