Семинар #8: Умные указатели. Домашнее задание.

Задача 1. test

```
#include <iostream>
#include <memory>

struct Cat
{
    Cat() {std::cout << "Constructor" << std::endl;}
    ~Cat() {std::cout << "Destructor" << std::endl;}
};

int main()
{
    Cat* raw = new Cat;
    std::unique_ptr<Cat> p(raw);
    std::unique_ptr<Cat> q(raw);
}
```

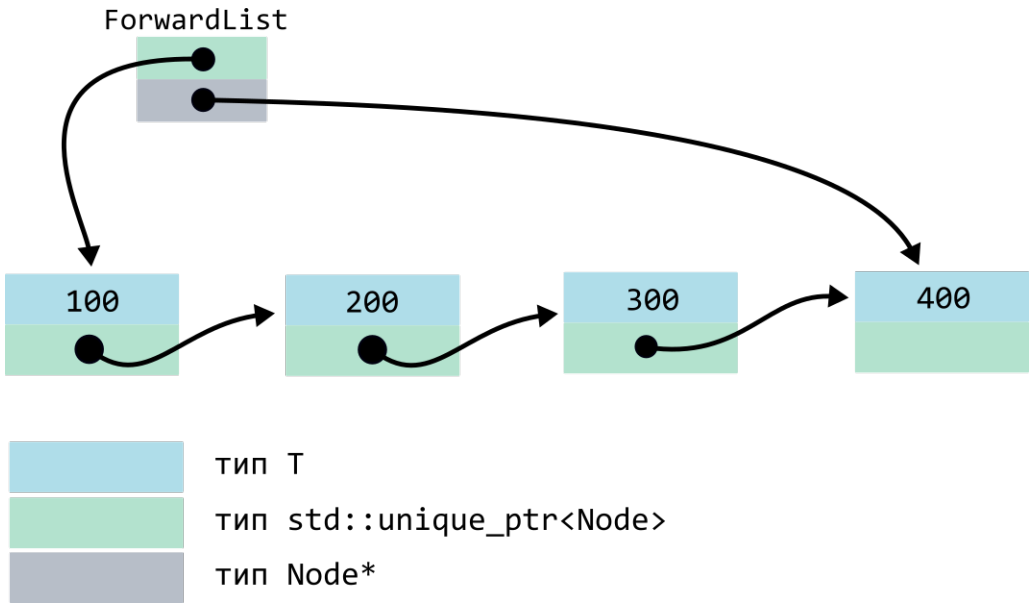
Задача 2. Односвязный список, используя `std::unique_ptr`

Создайте шаблонный класс `ForwardList<T>`, при этом, поле в узле списка, которое будет указывать на следующий узел должно иметь тип `std::unique_ptr`. Поля такого класса должны выглядеть так:

```
template <typename T>
class ForwardList
{
    struct Node
    {
        T value;
        std::unique_ptr<Node> next;
    };

    std::unique_ptr<Node> mpHead;
    Node* mpTail;
        ...
};
```

Схематическое строение объекта такого класса:



Вам нужно написать следующие методы данного класса:

- Конструктор по умолчанию.
- `void print()`
- `void push_front(T elem)`
- `void push_back(T elem)`
- `std::unique_ptr<T> pop_front()`
- `std::unique_ptr<T> pop_back()`
- `void clear()`
- `template <typename F> void foreach(F f)` – применяет функцию `f` к каждому элементу связного списка. Функция f должна принимать объект типа `T` по обычной ссылке.
- `void swap(ForwardList& fl)` - меняет местами содержимое данного связного списка и списка `fl`.
- `ForwardList copy()` - возвращает полную копию данного связного списка.