Оглавление

[Тема 1](#_Toc21070835)

[Преамбула 1](#_Toc21070836)

[Задачи 1](#_Toc21070837)

[Отчет 2](#_Toc21070838)

# Тема

Разработка класса префикс-функции. Переопределение операций, работа с динамической памятью.

# Преамбула

В этом задании вы разработаете класс префикс-функции PF, определенной в лекции 3. Получите практику в переопределении операций и управлении динамической памятью.

Прежде чем приступать к выполнению задания необходимо прочесть часть лекции относительно префикс-функции.

# Задачи

1. Начните новый консольный проект PrefixFunc. Подключите возможность консольного ввода-вывода.
2. Добавьте в проект класс PF. Его удобно сделать inline, т.е. полностью определить в заголовочном файле. Для этого нужно удалить из проекта файл PF.cpp и реализовать пока пустые конструктор и деструктор непосредственно в PF.h.
3. Согласно определению, префикс-функция – это массив целых …. Поэтому, в класс необходимо включить указатель на массив целых (int \* pv) и целочисленную переменную для хранения длины массива – n.  
     
   Как при этом должны измениться конструктор и деструктор?
4. Массив будет заполняться в методе void Init(const char \*s). Пусть метод пока просто выделяет память и заполняет поле n.   
     
   Подумайте, к чему приведет повторная инициализация? Как решить эту проблему?
5. Чтобы удобно было работать с объектами класса PF определите оператор преобразования типа PF => int. Прототип оператора - operator int () const. Пусть он просто возвращает длину массива.
6. Нам также понадобится оператор [], для получения конкретного значения функции по индексу. Прототип такого оператора будет иметь вид int operator [] (int i) const. Он просто возвращает значение i-того элемента массива.  
     
   Позаботьтесь чтобы индекс был в пределах массива, и если он выходит за диапазон, то оператор [] должен генерировать исключение. Это делается вызовом оператора throw out\_of\_range(“Index out of range”); Здесь out\_of\_range – класс исключения определенный в стандартной библиотеке <exception>.
7. Теперь нужно реализовать алгоритм вычисления префикс-функции, который имеет сложность O(N), где N – длина строки. Псевдокод алгоритма описан в лекции 3.   
     
   Завершите метод Init, разместив в нем реализацию алгоритма вычисления префикс-функции.
8. Создайте объект класса PF в main и проинициализируйте его строкой "rakaratrakarakarakatakarakara". Для вывода результата воспользуйтесь тем, что при преобразовании объекта в целое мы получаем длину массива, а для получения значения для конкретного индекса у нас есть оператор [].

Проверьте, что результат соответствует значению, приведенному в лекции (00001201234563456340000123456). Если это не так, то найдите и исправьте ошибки реализации.

1. В рамках курса мы не доказывали, что сложность алгоритма порядка O(N). Подтвердите это утверждение экспериментально! Для этого достаточно вычислить префикс-функцию для 4-5 строк различной длины, различающейся в несколько раз и подсчитать количество сравнений символов для каждой из них. Затем, отобразить значения графически.

В класс нужно добавить Счетчик сравнений и метод int CmpCount(), который возвращает значение счетчика. Не забудьте сбрасывать счетчик в начале Init.

Сгенерировать случайную строку из ‘a’ и ‘b’ длиной 100 можно следующим кодом:

const int N = 100;

char s[N]{0};

for (int i = 0; i < N - 1; i++)

s[i] = rand() % ('c' - 'a') + 'a';

Нужно попробовать строки не только из ‘a’ и ‘b’. Интересно, как зависит число операций, если набор символов увеличивается от 2 до, например, 32.

1. Подумайте, что будет происходить при копировании и присваивании объектов PF друг другу? Предложите (без реализации) путь решения возникающих проблем. Подумайте, а нужно ли вообще копирование префикс-функции?

# Отчет

Отчет должен содержать:

* Краткое описание применения префикс-функции для поиска подстроки в тексте.
* Ваши соображения (“на пальцах”, без использования кода) по реализации метода Init, ограничения на его повторный вызов, если они есть и как они устранены, если это было сделано.
* Пояснения к подсчету количества сравнений символов (без использования кода).
* Пояснение, почему цикл генерации z-строки выполняется до N – 1, а не до N, как делается в случае массива.
* Описание ваших экспериментов. Какова зависимость количества сравнений от длины строки? Как зависит результат от алфавита строки. Графики с результатами ваших экспериментов.
* Ваши соображения насчет копирования объектов разработанного класса PF.