Отчет по теме “Конечный автомат KMP. Применение алгоритма KMP для поиска вируса в файле. Работа с файлами”

Вариант 3

Алгоритм Кнута-Морриса-Пратта (KMP) используется для поиска подстроки(образца) в строке (тексте) за линейное время O(n). Основная идея – использование префикс-функции, кот-ая позволяет избежать повторного сравнения уже просмотренных символов при несовпадении (то есть, благодаря использованию префикс-функции мы сокращаем время использования алгоритма).

Ответ на вопрос из п.8.

Нужен ли деструктор в классе KMP\_StateMachine?

Да, он нужен, так как в конструкторе выделяется память под массив pattern и чтобы избежать утечки памяти – нам необходимо освободить её с помощью деструктора.

~KMP\_StateMachine() {

delete[] pattern; // освобождаем память

}

Как был определён размер массива vCode в main()? Почему это возможно?

Размер массива vCode был определён с помощью sizeof(vCode), который возвращает размер всего массива в байтах. VCode – это статический массив, поэтому sizeof возвращает корректно размер массива, а не указателя.

Как алгоритм перехода автомата связан с построением префикс-функции?  
Состояние автомата state соответсвует длине самого длинного совпадающего префикса образца с текущим окончанием текста.

Если происходит несовпадение символа, состояние изменяется на pf[state – 1] — значение префикс-функции для предыдущего состояния. Это позволяет откатиться к самому длинному префиксу, который мог бы продолжать совпадать, и продолжить сравнение без возврата по тексту.

Проанализировать реализацию автомата: допускаются ли пересечения вхождений?

В этом реализации пересечение вхождений **разрешено**, потому что после нахождения совпадения (IsFinal() == true) (раньше он был в конце метода Accept) состояние автомата сбрасывается в 0 (state = 0) (то есть в методе ScanFile в условии Isfinal никогда бы не сработало условие).

Если нужно разрешить пересечение – нужно оставить строку state = 0 в методе Accept().