**Заключение**

**Смоделировать двумя способами** новый тип «Список»:

1) на основе одномерного динамического массива.

2) на основе типизированного/двоичного файла.

1) В массив **добавлять** элементы несколько сложно поскольку при этом приходится копировать все элементы массива в новую память большего размера. В приведенной реализации память перевыделяется не каждый раз при добавлении нового элемента, а создается запас из 10 элементов (с конца), что позволяет повысить эффективность не менее чем в 10 раз. С файлом сложностей меньше, так как мы можем дозаписывать в файл информацию (в конец). Но при добавлении элементов в начало, приходится либо передвигать все элементы (в принципе как и с массивом, при наличие запаса), либо создавать новый файл (массив), с последующим выделением места под 1-й элемент. Также, при удалении элементов в файле, мы должны сдвигать все в начало (как в файле, так и в массиве), и чтобы не делать так постоянно, мы накапливаем 10 элементов сначала, а только потом сдвигаем, либо пересоздаем новую структуру без запаса сначала. Остальные функции имеют одинаковую сложность.

2) При работе с файлом, память выделяется (из кучи, сам файл хранится во внешней памяти) только на хранение максимум 1 файловой переменной (потока) (размером в 4 байта) и 2 элемента типа TInfo (72 байта каждый).

При работе с массивом, мы храним весь массив (включая запас справа и слева) в куче, из-за чего он занимает большое место.

3) Но время, затраченное на работу с файлом, больше, по сравнению с временем затраченным устройством для работы с массивом, хоть это и не очень заметно.

**Вывод:** При работе с файлом, мы экономим большое кол-во оперативной памяти, но жертвуем производительностью и скоростью работы. При использовании массива с запасами, мы работаем намного быстрее, чем с файлом. Без использования запаса в массиве, мы бы при каждом добавлении и удалении должны были бы выделять новую память под массив, что очень сильно отразилось бы на его эффективности.