|  |  |
| --- | --- |
| Дек:  3  50  2  3 | Дек:  50  2 |
| Дек:  3  50  1  2  30 | Дек:  50  1  2 |
| Дек: | Дек пуст! |

7. Вывод по работе

При реализации стека, дека и очереди в виде динамического массива необходимо резервировать длину массива, длина которого равна количеству элементов стека, дека и очереди соответственно, что приводит к неэффективному использованию памяти, а также неудобствам при удалении и добавлении элементов: операции уменьшения и увеличения длины массива удаляют и добавляют память для элемента с конца массива, что вызывает неудобства при необходимости добавить или удалить элемент в «середину» структуры. Для того, чтобы удалить элемент из «середины» структуры, нужно сдвинуть элементы структуры так, чтобы элемент, который нужно удалить, находился на последнем месте в массиве. Для того, чтобы добавить элемент, нужно увеличить длину массива, а затем сдвинуть все элементы массива вправо на одну позицию и присвоить нужной ячейке нужное значение. По моему мнению, плюсом реализации этих структур на основе массива является отсутствие связей ^.next и ^.prev (в случае дека), что приводит к небольшому упрощению при работе с ними.

Рассмотрим плюсы и недостатки использования различных структур на основе списка:

1. Стек. Плюсы: процедуры добавления и удаления элемента распространяются на верхний элемент. Недостатки: для работы с элементами в середине стека, для получении информации о последнем элементе нужно использовать перекладывание Top to Top.
2. Очередь. Плюсы: при работе с элементами в середине перекладывание элементов Top to Bottom используется до конца в буферную очередь (в стеке до нужного элемента), а потом исходной очереди присваивается значение буферной. Недостатки: для работы с элементами в середине так же нужно использовать перекладывание.

Примечание: для удобной работы с очередью на основе списка лучше использовать указатели на начало и конец очереди.

1. Дек. Плюсы: добавление и удаление элементов реализуется с двух сторон, что добавляет возможностей при работе со структурой. Недостатки: реализуется на основе двусвязного списка ( указатели ^.next и ^.prev ), что усложняет работу с элементами.

Сравнение списка и массива.

Доступ. Связанные списки позволяют использовать только последовательный способ доступа к элементам, следовательно, алгоритмические сложности – порядок О(n).

Массив предоставляет произвольный доступ к элементам за счет своей индексации, порядок сложности О(1).

Хранение. Связанные списки требуют дополнительных переменных для хранения ссылок на элементы (1 или 2). Это делает их непрактичными с точки зрения памяти для списков небольших элементов данных, таких как Char или Boolean.

Массив не требует дополнительных переменных для хранения ссылок, доступ осуществляется через индексы.

Размер. Связанные списки динамичны по своей природе, количество элементов не нужно фиксировать, их размер ограничен лишь объемом памяти.

Размер массива всегда нужно фиксировать, при выходе элемента за границу отведенной памяти необходимо перевыделять память, в результате чего элементы массива копируются в новый, при этом операция вставки элемента выполняется относительно долго.

Удаление и добавление элемента. В списках удаление/добавление элемента осуществяется путем перестановки указателей, при этом сами данные не копируются.

В массиве требуется копирование всех последующих элементов для их сдвига.

Вывод. Использование связанного списка выгодно, когда количество элементов постоянно растет и сокращается, а использование динамического массива, когда известно точное количество элементов и важен поиск элементов по индексу.