МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Нижегородский государственный технический университет им. Р.Е. Алексеева»

Кафедра "Информационные радиосистемы"

Реализация динамических структур на массиве

Методические указания к лабораторной работе № 1
по дисциплине «Информационные технологии» для студентов
направления подготовки бакалавра
210400 «Радиотехника»
дневной формы обучения

Нижний Новгород 2012

Составили**:** Е.Н.Приблудова, С.Б.Сидоров, М.В.Уханов

УДК 621.325.5-181.4

Реализация динамических структур на массиве: метод. указания к лаб. работе по дисциплине «Информационные технологии» для студентов направления подготовки бакалавра 210400 «Радиотехника» дневной формы обучения / НГТУ; сост.: Е.Н.Приблудова, С.Б.Сидоров, М.В.Уханов. Н.Новгород, 2012, 10 с.

Изложены краткие сведения о динамических структурах данных, способах их реализации на массиве. Сформулирован порядок выполнения лабораторной работы. Приведены контрольные вопросы для самопроверки.

Научный редактор А.Г.Рындык

Редактор Э.Б.Абросимова

Подп. к печ. Формат 60x84 1/16. Бумага газетная.
Печать офсетная. Печ.л. 0,75. Уч.-изд.л. 0,5 . Тираж Заказ .

Нижегородский государственный технический университет им. Р.Е.Алексеева
Типография НГТУ. 603950, Н.Новгород, ул.Минина, 24.

1. Цель работы

Практическое изучение методики непрерывной реализации динамической структуры на массиве.

2. Краткие сведения

Под термином *структура данных* будем понимать организацию данных, т.е. некоторый способ объединения этих данных. Чтобы правильно и эффективно использовать имеющиеся в распоряжении разработчика средства, важно добиться хорошего понимания структурных отношений, существующих между данными, способов представления таких структур в машине и методов работы с ними. Их знание дает возможность создать модель, более адекватную решаемой задаче.

Будем рассматривать структуры данных (сложноорганизованные типы данных), которые представляют собой конечную совокупность некоторых объектов (элементов) одного и того же типа *E*. Между собой структуры различаются тем, как именно они работают с элементами, т.е. отличаются набором операций над элементами и способом упорядоченности элементов, или, говоря иначе, заданным *отношением следования между элементами*.

Кроме этого, из всевозможных структур, которые удовлетворяют приведенному выше описанию, будем выделять *динамические структуры данных*. Введенный термин *динамические структуры* означает, что число элементов в них может изменяться в процессе работы.

Наиболее часто используемыми динамическими структурами являются: стек, динамический вектор, очередь, дек, список, последовательность, множество, дерево, двоичное дерево.

Для каждой динамической структуры кроме отношения следования между элементами определен и набор операций, выполняемых над структурой (не над элементами, а именно над структурой).

Как правило, динамические структуры не являются предопределенными типами данных в языках программирования. Отсюда вытекает необходимость реализации поддержки требуемых динамических структур. Эта реализация должна быть проведена с использованием средств и возможностей конкретного языка программирования. Часто рассматривается задача реализации одних динамических структур на базе других.

Для представления динамической структуры может использоваться непрерывная реализация. В этом случае объекты типа *E*, входящие в динамическую структуру, будут располагаться последовательно в соответствующих элементах массива. Элементы массива также будут иметь тип *E*. Поскольку массив является статической структурой данных, у которой количество элементов

должно задаваться при ее описании, то реализованная таким способом динамическая структура может иметь ограниченное количество элементов. Оно определяется количеством элементов массива.

Для непрерывной реализации динамической структуры характерно наличие переменных (целочисленных), значения которых определяют положение структуры данных в массиве. Эти переменные содержат индексы элементов массива, в которых располагаются первый и последний элементы структуры. На рис. 1 приведена графическая иллюстрация общей схемы непрерывной реализации динамической структуры на массиве. Переменная **First** содержит индекс элемента массива, предшествующего первому элементу структуры, а **Last** - индекс последнего элемента. Индексы **First** и **Last** могут принимать значения от -1 до *N*-1. Если **First = Last = -**1, то динамическая структура не содержит элементов.

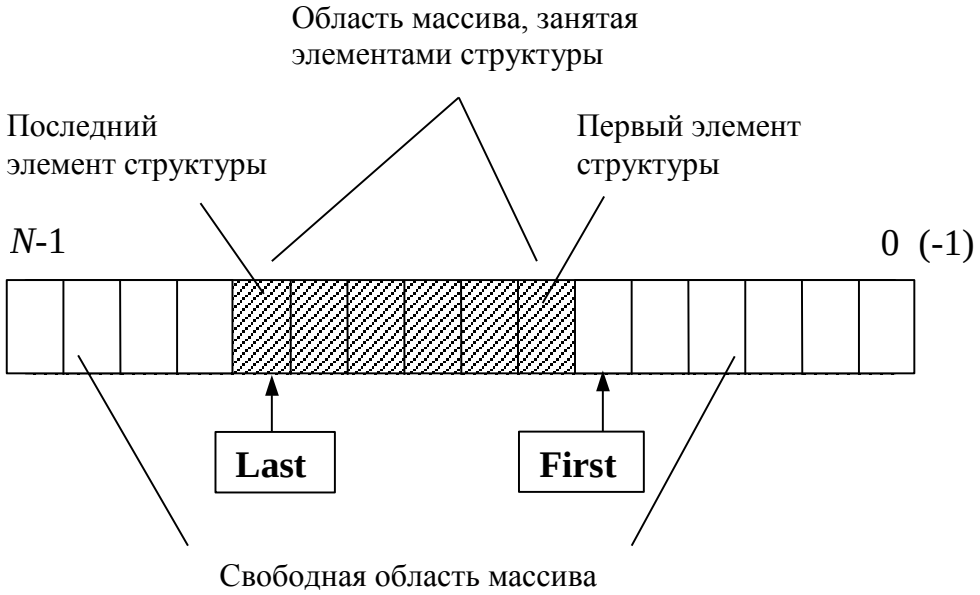


Рис. 1

Рассмотрим некоторые динамические структуры, в частности:

- стек;
- очередь;
- дек.

Основные отличия приведенных динамических структур состоят в отношении следования элементов внутри структуры и в способах добавления или удаления элементов.

Стеком называется упорядоченный набор некоторого переменного числа объектов (возможно, нулевого), работающий по правилу, – "последний пришел, первый вышел". Часто стек определяется аббревиатурой LIFO – Last In First Out. *Вершина стека* – это индекс элемента, записанного последним.

Для стека определяются следующие операции:

- добавление одного элемента;
- проверка, определяющая, пуст ли стек;
- извлечение одного элемента.

Алгоритм добавления одного элемента в стек:

- увеличить индекс;
- добавить один элемент.

Проверяя пуст ли стек, необходимо выяснить равенство индекса –1.

Алгоритм извлечения одного элемента из стека:

- извлечь один элемент;
- уменьшить индекс.

На рис. 2 представлена графическая иллюстрация работы стека:

- стек пуст (*а*);
- добавление одного элемента (*б*);
- добавление еще одного элемента (*в*);
- извлечение одного элемента (*г*);
- извлечение еще одного элемента, стек пуст (*д*).

Очередью называется упорядоченный набор некоторого переменного числа объектов, работающий по принципу, – "первый пришел, первый вышел". Часто очередь определяется аббревиатурой FIFO – First In First Out.

Рассмотрим идеи реализации динамической структуры типа *очередь* на массиве. Поскольку операции над очередью производятся с двух концов, то необходимо рассмотреть две переменные **First** и **Last**.

Для очереди определяются следующие операции:

- добавление одного элемента в конец очереди;
- проверка, определяющая: является ли очередь пустой;
- извлечение одного элемента из начала очереди.

Алгоритм добавления одного элемента в очередь:

- увеличить значение **Last**;
- добавить один элемент.

Алгоритм извлечения одного элемента из очереди:

- увеличить значение **First**;
- извлечь один элемент.

На рис. 3 представлена графическая иллюстрация работы очереди (переменные **First** и **Last**, обведенные пунктирной линией, обозначают предыдущие положения индексов):

- очередь пустая (*а*);
- добавление одного элемента (*б*);
- добавление еще одного элемента (*в*);
- извлечение одного элемента (*г*);
- извлечение еще одного элемента, очередь пустая (*д*);
- добавление двух элементов (по одному) на массиве, свернутом в кольцо (*е*).

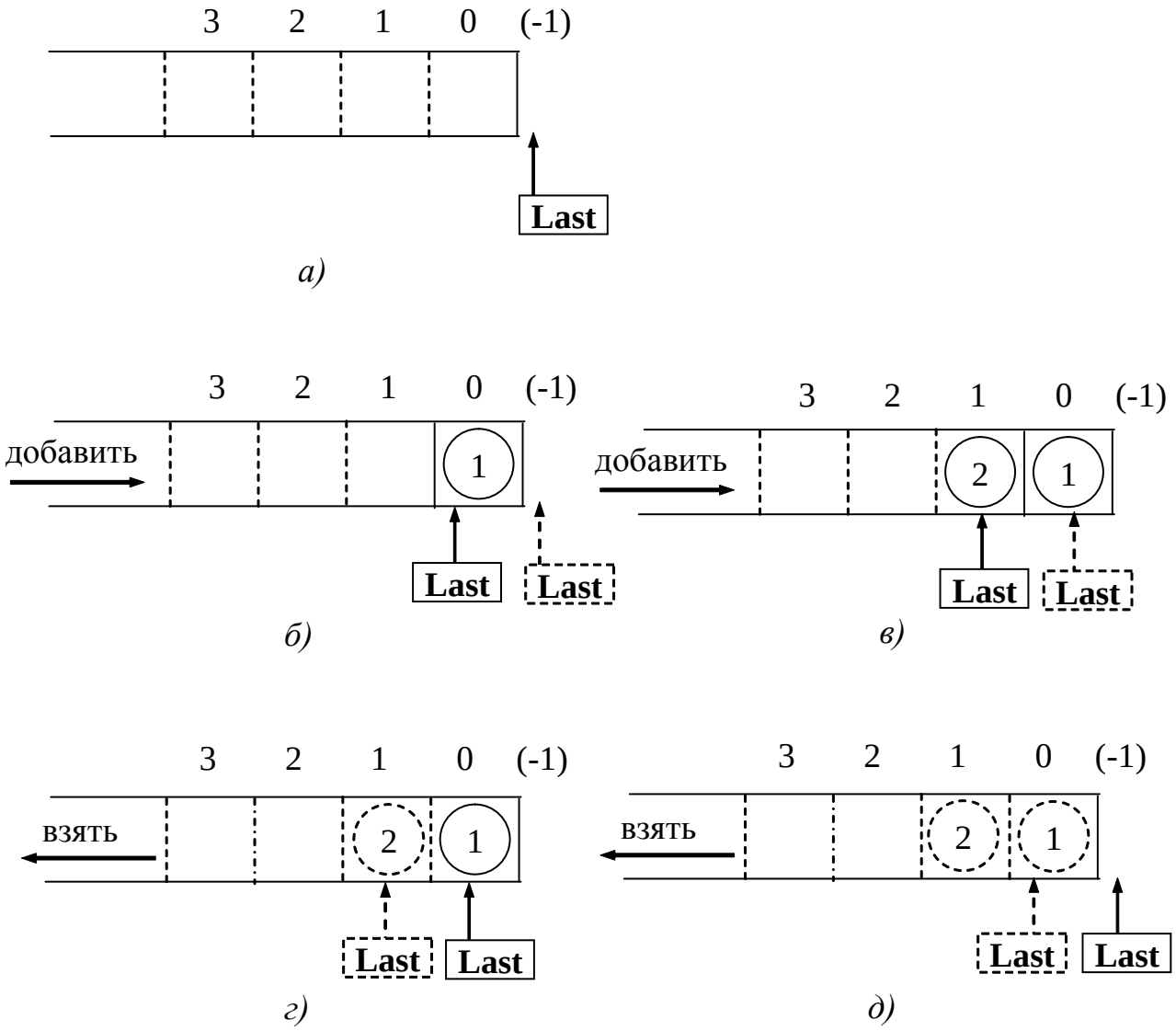


Рис. 2

При непрерывной реализации очереди необходимо решить ряд проблем.

Может возникнуть ситуация, при которой элементы очереди занимают весь массив. Тогда занесение в очередь еще одного элемента не может быть произведено. Для отслеживания этого случая необходимо уметь определять, имеется ли свободное место для занесения в очередь нового элемента.

Особая ситуация возникает при попытке занесения элемента в очередь, показанную на рис 3, *е*. В данном случае необходимо добавить сначала один элемент, затем другой в очередь. Но в конце очереди свободно место только под один элемент. В то же время из рисунка видно, что в массиве имеются свободные места для новых элементов. Возможно несколько подходов к решению данной проблемы.

Прежде всего, можно сдвинуть все элементы очереди в начало массива. Таким образом, слева от последнего элемента в очереди появится свободное место, куда теперь можно занести новый элемент. Данный способ реализуется

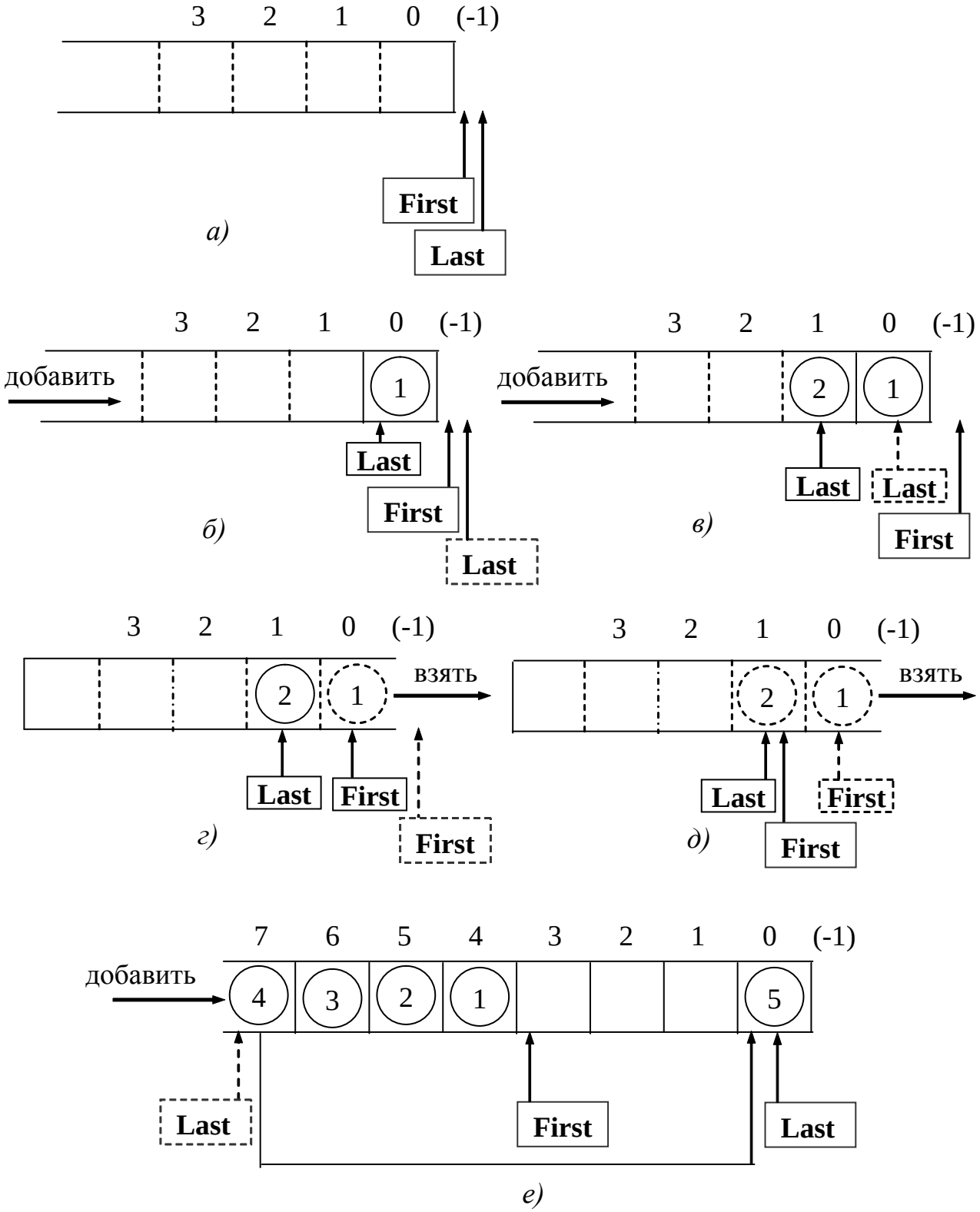


Рис. 3

просто, но при этом выполняется массовая операция перемещения элементов внутри массива. При больших размерах очереди такой способ неприемлем.

Второй способ лишен указанного недостатка. Он основывается на идеи сворачивания массива в кольцо (рис. 3, *е*). Таким образом, для добавления второго элемента необходимо использовать свободное место в начале очереди. При этом выполняется соединение первого и последнего элементов массива. В этом случае следующим после крайнего слева элемента массива становится первый справа элемент массива. Если k – индекс элемента массива, а N – количество элементов в массиве, то индекс следующего за ним можно определить по формуле

Следующий = (k+1) остаток от деления на N.

Таким образом, при добавлении нового элемента индекс элемента массива, куда необходимо произвести запись, определяется из приведенной формулы. Аналогичным способом производится и извлечение элемента из очереди. При этом новое значение переменной **First** определяется по той же самой формуле.

Обобщением очереди является динамическая структура ***дек***. В такой динамической структуре добавлять и извлекать элементы можно с любой стороны. При организации динамической структуры *дек* необходимо переменные **First** и **Last** расположить в середине массива рис. 4, *а*. В данном примере число элементов массива – четное, поэтому за середину может выбираться элемент с индексом 3 или 4. В нашем случае за середину выбран элемент с индексом 3. На рис. 4 представлена графическая иллюстрация работы дека.

Для дека определены следующие операции:

- добавление элемента в начало;
- добавление элемента в конец;
- проверка, определяющая: является ли дек пустым;
- извлечение элемента из начала;
- извлечение элемента из конца.

Алгоритм добавления одного элемента в начало дека:

- добавить один элемент с индексом **First;**
- уменьшить индекс **First.**

Алгоритм добавления одного элемента в конец дека:

- увеличить индекс **Last**;
- добавить один элемент с индексом **Last.**

Алгоритм извлечения одного элемента с начала дека:

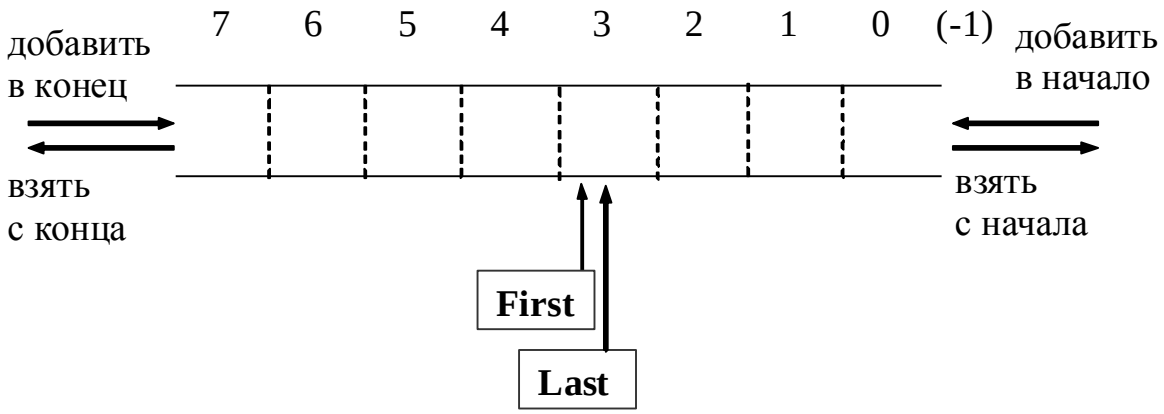
- увеличить индекс **First;**
- извлечь один элемент с индексом **First**;

Алгоритм извлечения одного элемента с конца дека:

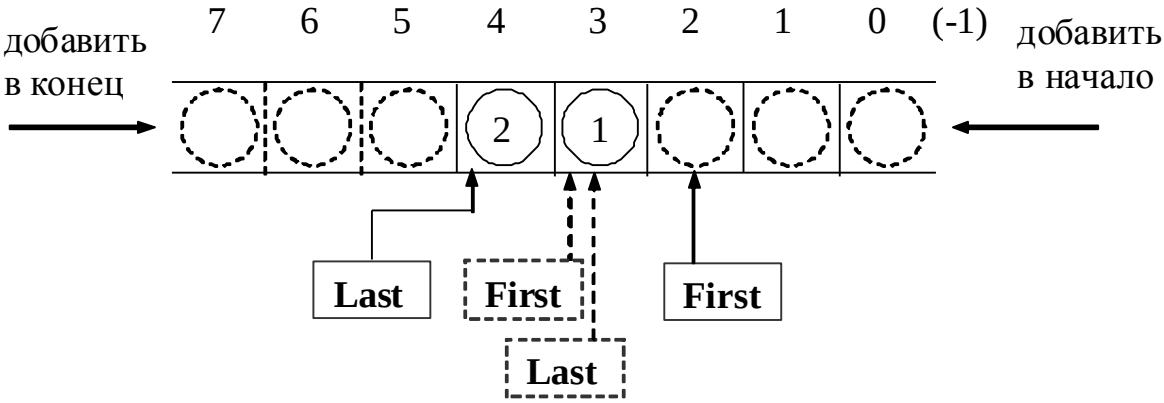
- извлечь один элемент с индексом **Last**;
- уменьшить индекс **Last.**

На рис. 4 представлена графическая иллюстрация работы дека:

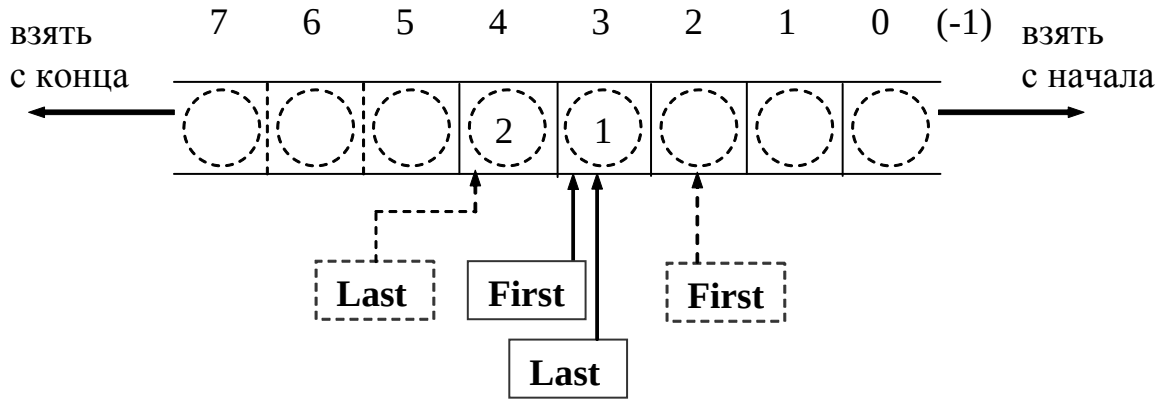
- дек пуст (*а*);
- добавление одного элемента в начало и в конец (*б*);
- извлечение одного элемента из начала и из конца, дек пуст (*в*).



а)



б)



в)

Рис. 4

3. Порядок выполнения лабораторной работы

1. Подготовка к работе.

Изучите теоретическую часть по данной теме. Ознакомьтесь с вариантом задания на лабораторную работу, определяемым преподавателем.

2. Выполнение работы.

Выполнение работы состоит в следующем:

- анализ варианта задания;
- разработка алгоритма решения задачи, определение структуры программы и данных, используемых в ней;
- представление разработанного алгоритма преподавателю;
- получение программной реализации на массиве одной из динамических структур (все операции оформить в виде функций);
- отладка полученной программы, реализующей основной набор операций с заданной динамической структурой;
- внесение в программу изменений, предложенных преподавателем.

3. Отчетность лабораторной работы.

Отчет по лабораторной работе должен содержать:

- исходные данные;
- алгоритм, составленный с помощью блок-схемы;
- представление на экране исходного текста программы и результата ее работы, текст программы должен иметь комментарии.

4. Контрольные вопросы

1. Как Вы понимаете термин «структура данных»?
2. Что такое «динамическая структура данных»?
3. Приведите примеры динамических структур.
4. В чем заключается задача реализации одних динамических структур на базе других?
5. Что такое непрерывная реализация динамической структуры?
6. Приведите примеры реальных понятий, процессов, модель которых может быть представлена с использованием динамических структур.

5. Список рекомендуемой литературы

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня: учебник / Т.А.Павловская.- СПб.: Питер, 2005.
2. Шилдт, Г. Полный справочник по С, 4-е издание.: [пер. с анг.] / Г.Шилдт.- М.: Вильямс, 2005.
3. Демидович Е. М. Основы алгоритмизации и программирования. Язык Си: Учебное пособие / Е.М.Демидович.- БХВ-Петербург, 2008.