**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное ГОСУДАРСТВЕННОЕ бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики

и вычислительной техники

Кафедра информационной

безопасности

**ОТЧЕТ**

**По Лабораторной работе №3**

Выполнил:

студент гр. БИс-32

Фомин Е.В.

Касьянов Р.А.

Семёнов Н.М.

Волгин М.А.

Проверил:

Ассистент кафедры ИБ

Ситников И.В.

Йошкар-Ола

2017

**Ход работы**

**Цель работы:** Написать свою реализацию коллекций Array, Queue, Stack.

**Требования к программе:**

1. Разберитесь, чем отличаются коллекции друг от друга, какие методы реализует каждая из коллекций. Разобраться как используются универсальные шаблоны.
2. Коллекции могут принимать для хранения различные типы данных. Реализовать данный функционал в своей программе.
3. Все три коллекции должны наследоваться от одного базового абстрактного класса. Подумайте какие методы можно оставить в абстрактном классе (общие для всех классов-наследников).
4. Реализовать методы, которые отличают коллекции друг от друга (например, методы **Pop/Push**для коллекции Stackи т.д.).
5. Реализовать несколько конструкторов с различными принимаемыми параметрами.

**Теоретические сведения:**

**Стек -** динамическая структура данных, которая работает по принципу "Последним вошел - первым вышел". По английски LIFO (last-in, first-out). Как можно видеть мы добавляем один элемент с помощью метода Push, и потом с помощью метода Pop мы его вытаскиваем(Рисунок 1)

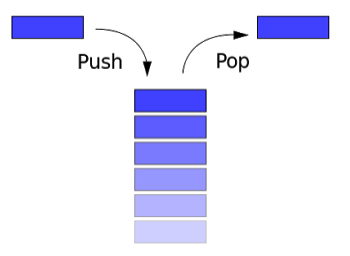


Рисунок 1

Применение:

* Стек можно использовать для того, чтобы избавится от рекурсии, так идея вызова рекурсивных функций сама использует стек. Адрес возврата и локальные переменные рекурсивной функции записываются в стек, благодаря чему каждый следующий рекурсивный вызов этой функции пользуется своим набором локальных переменных и за счёт этого работает корректно. Единственный минус такого использования в том, что очень **хорошо отъедает память** :D, поэтому надо избегать программ, которая допускает большой глубины рекурсии.
* Стек используется при парсинге HTML/XML деревьев.
* Существует **стек вызовов** (call stack), в который заносится **информация о возврате из функции**. Например при вызове функции A() из функции main, в стек вызовов заносится адрес возврата, то есть адрес следующей функции, в которую должно передатся управление, например у нас функция (не рекурсивна) A() {return 0;}, то адрес возврата у нас будет адрес функции main(). К тому же заносится **информация о локальных переменных** и **значения параметров функции**.
  + Например, у нас есть функция main из нее вызывается функция A(), из A() вызывается B(), то стек у нас будет вот такой: Head-> **[B(),A(),main()]**. B()-вершина стека, ниже будет А(), и на дне стека main(). При возврате из B() управление передастся к A(), при возврате из A() управление передастся к main().
* Также стек используется в калькуляторе. Например вот из этого:
  + **Input:** (((8 + 1) - (7 - 4)) / (11 - 9)) Превратили в вот это с помощью стека (из инфиксной в обратную польскую)

**Output:** 8 1 + 7 4 - - 11 9 - /

**О́чередь** — абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел» (FIFO, First In — First Out). Добавление элемента (принято обозначать словом enqueue — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом dequeue — убрать из очереди), при этом выбранный элемент из очереди удаляется.(Рисунок 2)

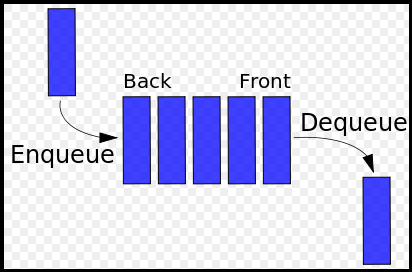


Рисунок 2.

Очередь в программировании используется, как и в реальной жизни, когда нужно совершить какие-то действия в порядке их поступления, выполнив их последовательно. Примером может служить организация событий в Windows. Когда пользователь оказывает какое-то действие на приложение, то в приложении не вызывается соответствующая процедура (ведь в этот момент приложение может совершать другие действия), а ему присылается сообщение, содержащее информацию о совершенном действии, это сообщение ставится в очередь, и только когда будут обработаны сообщения, пришедшие ранее, приложение выполнит необходимое действие.

Клавиатурный буфер BIOS организован в виде кольцевого массива, обычно длиной в 16 машинных слов, и двух указателей: на следующий элемент в нём и на первый незанятый элемент.

**Массив** (в некоторых языках программирования также таблица, ряд, матрица) — структура данных в виде набора компонентов (элементов массива), расположенных в памяти непосредственно друг за другом. За счёт этого, в отличие от, например, связного списка, вычислительная сложность для доступа к конкретному элементу по индексу константна. Массив реализует абстрактный тип данных список.

**Размерность** массива — это количество индексов, необходимое для однозначной адресации элемента в рамках массива. **Форма** или **структура массива** — сведения о количестве размерностей и размере (протяжённость) массива для каждой из размерностей; может быть представлена одномерным массивом.

**Достоинства**

* лёгкость вычисления адреса элемента по его индексу (поскольку элементы массива располагаются один за другим)
* одинаковое время доступа ко всем элементам
* малый размер элементов: они состоят только из информационного поля.

**Недостатки**

* для статического массива — отсутствие динамики, невозможность удаления или добавления элемента без сдвига других
* для динамического и/или гетерогенного массива — более низкое (по сравнению с обычным статическим) быстродействие и дополнительные накладные расходы на поддержку динамических свойств и/или гетерогенности.
* при работе с массивом в стиле C (с указателями) и при отсутствии дополнительных средств контроля — угроза выхода за границы массива и повреждения данных.

# Ссылка на GitHub: <https://github.com/KPSS322/Pna9vy3>

# Вывод

В ходе выполнения лабораторной работы, были реализованы коллекции Array, Stack и Queue и отдельный абстрактный класс, в котором хранятся переменные массива, размера, метки и 4 метода: проверка на пустоту, определение размера, вспомогательный метод для масштабирования, определение количества элементов в списке.