

**БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**Ханты-Мансийского автономного округа–Югры**

**«Сургутский государственный университет»**

Политехнический институт

Кафедра Автоматики и компьютерный систем

**Пояснительная записка**

к курсовому проекту

по дисциплине

Теория разработки программного обеспечения

Мобильное приложение для фиксации и отправки нарушений ПДД

Выполнил: студенты группы 609-81

\_\_\_\_\_\_\_\_Алексеев А.В.

\_\_\_\_\_\_\_\_Сиренко Н.В.

Проверил:

\_\_\_\_\_\_\_\_Емельянов С.Н.

**Сургут**

**2021**

# Техническое Задание

# Аннотация

В работе выполнен анализ существующих решений по аналогичной теме, после чего выстроен архитектура и дизайн мобильного приложений с учётом лучших качеств аналогичных приложений. Проведены проверки корректной работы приложения и сделаны выводы о дальнейшем его развитии и возможных перспективах.

НАПИСАТЬ СКОЛЬКО СТРАНИЦ И ЗАМЕНИТЬ ЛИТЕРАТУРУ

Работа состоит из 63 страниц, содержит 4 литературных источника.

Содержание

[Задание 2](#_Toc61896700)

[Аннотация 3](#_Toc61896701)

[Введение 6](#_Toc61896702)

[1 Анализ задачи 7](#_Toc61896703)

[1.1 Описание предметной области 7](#_Toc61896704)

[1.2 Обзор существующих решений 7](#_Toc61896705)

[1.2.1 Первая реализация 7](#_Toc61896706)

[1.2.2 Вторая реализация 9](#_Toc61896707)

[1.3 Возможное применение очереди с приоритетами 10](#_Toc61896708)

[2 Проектирование 12](#_Toc61896709)

[2.1 Техническое задани 12](#_Toc61896710)

[2.2 Проектирование класса очереди 13](#_Toc61896711)

[3 Кодирование и отладка 16](#_Toc61896712)

[3.1 Использование класса 16](#_Toc61896713)

[3.2 Основные методы класса QueuePriorityDelete 16](#_Toc61896714)

[4 Тестирование 18](#_Toc61896715)

[4.1 работоспособность очереди с приоритетом 18](#_Toc61896716)

[4.2 Тестирование оценки времени 20](#_Toc61896717)

[4.3 Оценка асимптотической сложности реализованных алгоритмов 25](#_Toc61896718)

[5 Сопровождение 27](#_Toc61896719)

[Заключение 28](#_Toc61896720)

[Список использованной литературы 28](#_Toc61896721)

[Приложения 29](#_Toc61896722)

[Приложение А. Полные листинги файлов 29](#_Toc61896723)

[А.1 Файл IItem.cs 29](#_Toc61896724)

[А.2 Файл Item.cs 29](#_Toc61896725)

[А.3 Файл IQueue.CS 29](#_Toc61896726)

[А.4 Файл QueuePriorityDelete.cs 30](#_Toc61896727)

[А.5 Файл BaseRelayCommand.CS 31](#_Toc61896728)

[А.5 Файл RelayCommand.cs 31](#_Toc61896729)

[А.6 Файл MainViewModel.cs 32](#_Toc61896730)

[А.7 Файл MainWindow.xaml 34](#_Toc61896731)

[А.8 Файл MainWindow.xaml.cs 36](#_Toc61896732)

[Приложение Б. Таблицы с данными тестов 38](#_Toc61896733)

[Б.1 Результаты тестов удаления с 50000 до 500000 с шагом 50000 38](#_Toc61896734)

[Б.2 Результаты тестов удаления с 50000 до 5000000 с шагом 50000 38](#_Toc61896735)

[Б.3 Результаты тестов удаления с 50000 до 50000000 с шагом 50000 40](#_Toc61896736)

# Введение

Этот документ представляет из себя пояснительную записку к программному продукту, требования для которого заданы в задании. Цель этого курсового проекта состоит в написании на языке С# и Python программного обеспечения и сопровождения к нему. Работа разделена на этапы, каждый из которых ниже подробно описан и представлен с примерами:

1. На первом этапе были проанализированы цель и задачи курсового проекта, собраны различные материалы из интернета, после чего подготовлено основное назначение программного продукта.

2. На этапе проектирования показывается разработанная архитектура приложения, описываются классы и концептуальные решения, удовлетворяющие поставленной задаче.

3. Используя модели, составленные на прошлом этапе, во время кодирования реализованы полностью работоспособный и рабочий код, который соответствует всем требованиям.

4. На последнем этапе, но не по значимости, проверяются все функции во всех возможных случаях и выявляется наличие ошибок и плохо проработанных частей кода, при обнаружении ошибок они сразу исправляются. В сопровождении описано, как работать и взаимодействовать с готовым программным продуктом.

# 1 **Анализ задачи**

## **Описание предметной области**

Настоящие Правила дорожного движения 2021 года устанавливают единый порядок дорожного движения на всей территории Российской Федерации(РФ). Другие нормативные акты, касающиеся дорожного движения, должны основываться на требованиях Правил и не противоречить им.

Так как приложение направлено на фиксацию нарушений ПДД связанных с парковкой, то ниже приведен список, когда и где запрещено парковать транспортное средство.

Остановка запрещается:

* на трамвайных путях, а также в непосредственной близости от них, если это создаст помехи движению трамваев;
* на железнодорожных переездах, в тоннелях, а также на эстакадах, мостах, путепроводах (если для движения в данном направлении имеется менее трех полос) и под ними;
* в местах, где расстояние между сплошной линией разметки (кроме обозначающей край проезжей части), разделительной полосой или противоположным краем проезжей части и остановившимся транспортным средством менее 3 м;
* на пешеходных переходах и ближе 5 м перед ними;
* на проезжей части вблизи опасных поворотов и выпуклых переломов продольного профиля дороги при видимости дороги менее 100 м хотя бы в одном направлении;
* на пересечении проезжих частей и ближе 5 м от края пересекаемой проезжей части, за исключением стороны напротив бокового проезда трехсторонних пересечений (перекрестков), имеющих сплошную линию разметки или разделительную полосу;
* ближе 15 метров от мест остановки маршрутных транспортных средств или стоянки легковых такси, обозначенных разметкой, а при ее отсутствии - от указателя места остановки маршрутных транспортных средств или стоянки легковых такси (кроме остановки для посадки и высадки пассажиров, если это не создаст помех движению маршрутных транспортных средств или транспортных средств, используемых в качестве легкового такси);
* в местах, где транспортное средство закроет от других водителей сигналы светофора, дорожные знаки, или сделает невозможным движение (въезд или выезд) других транспортных средств (в том числе на велосипедных или велопешеходных дорожках, а также ближе 5 м от пересечения велосипедной или велопешеходной дорожки с проезжей частью), или создаст помехи для движения пешеходов (в том числе в местах сопряжения проезжей части и тротуара в одном уровне, предназначенных для движения маломобильных граждан);
* на полосе для велосипедистов.

## 1.2 Обзор существующих решений

### 1.2.1 Первая реализация

Первая рассматриваемая реализация очереди с приоритетами была написана на С++. В этой реализации была особенность, что изначально есть какой-то заданный предел количества элементов очереди. Этот предел задаётся в конструкторе класса. Сама реализация – это очередь с приоритетным исключением. Такой вывод был сделан, потому что приоритет элементов учитывался, когда элемент вынимался из очереди, а не при добавлении элемента в очередь. Это можно увидеть на рисунке 1.

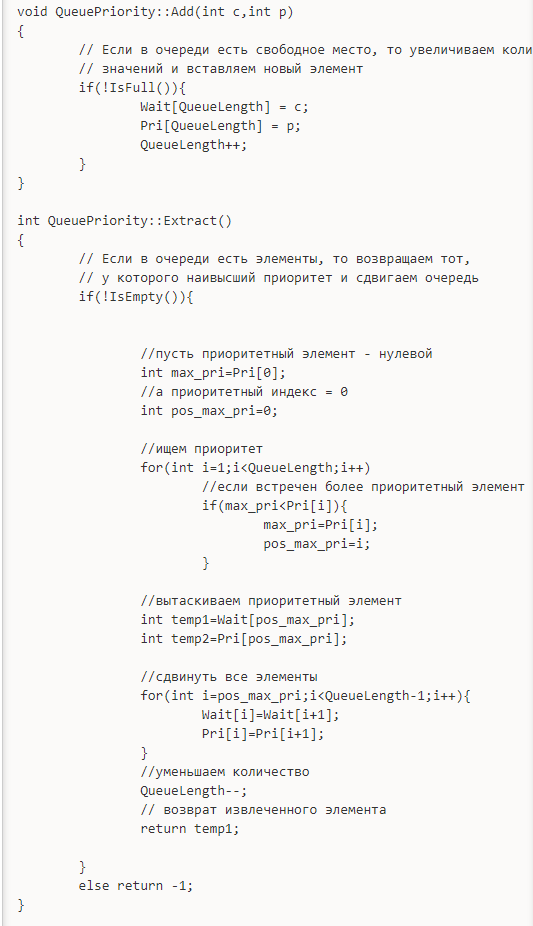


Рисунок 1 – Пример реализации добавления и извлечение из очереди

### 1.2.2 Вторая реализация

Вторая рассматриваемая реализация была, как и первая, сделана на С++. В отличии от первой реализации, во второй присутствует динамическое расширение и уменьшение массива, который хранит элементы в очереди. Это можно считать преимуществом на фоне первой реализации. Вторая реализация — это пример очереди с приоритетным добавлением. Сделать такой вывод можно посмотреть на рисунках 2-3, где видно, что в методе Push учитывается приоритет добавляемого элемента.

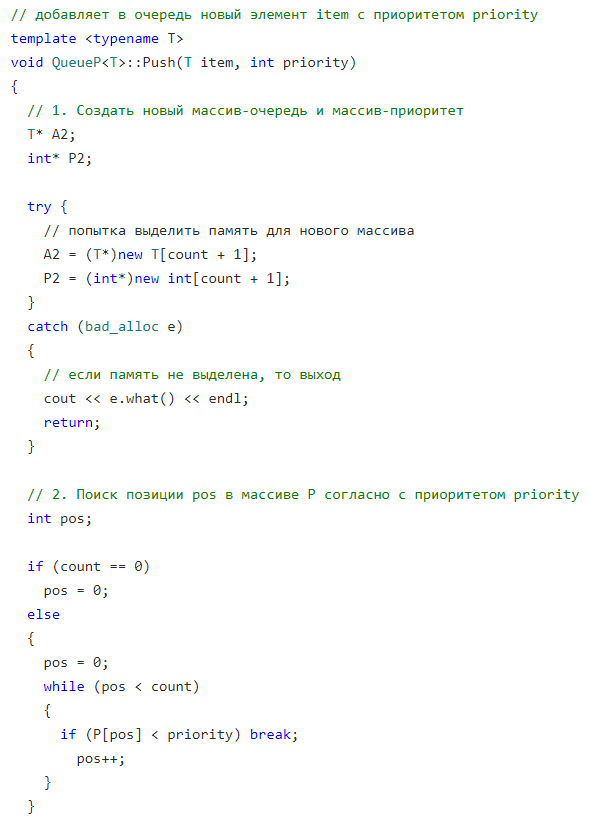


Рисунок 2 – Реализация добавления элемента из второго примера

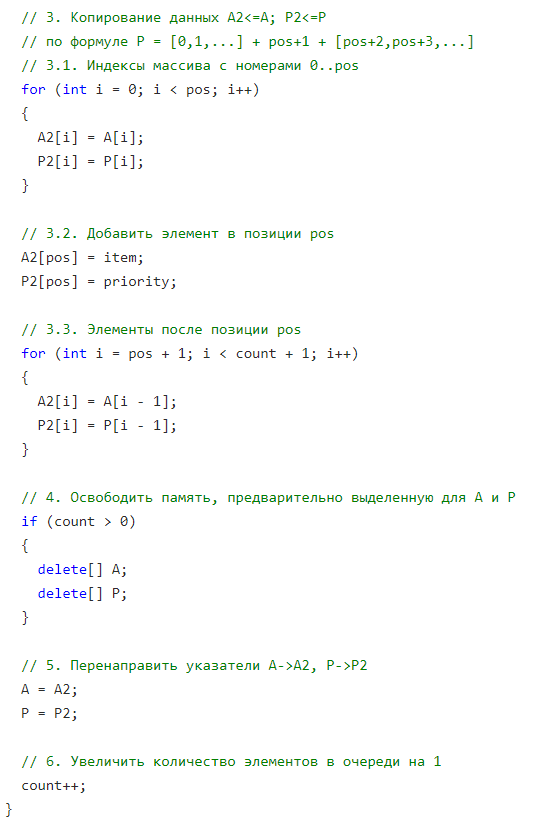


Рисунок 3 – Реализация добавления элемента из второго примера

## 1.3 Возможное применение очереди с приоритетами

Первое место, где будет полезна очередь с приоритетами и активно используется в настоящее время это процессы в различных операционных системах. Для примера возьмём операционную систему Windows. Чтобы посмотреть какие приоритеты могут быть в этой операционной системе необходимо открыть диспетчер задач, где нужно выбрать любой процесс, выбрать в всплывающем меню «подробно». В новой вкладке, снова открывает всплывающее меню, чтобы увидеть раздел «Задать приоритет». На рисунке 4 можно увидеть приоритеты для процессов в операционной системе Windows.

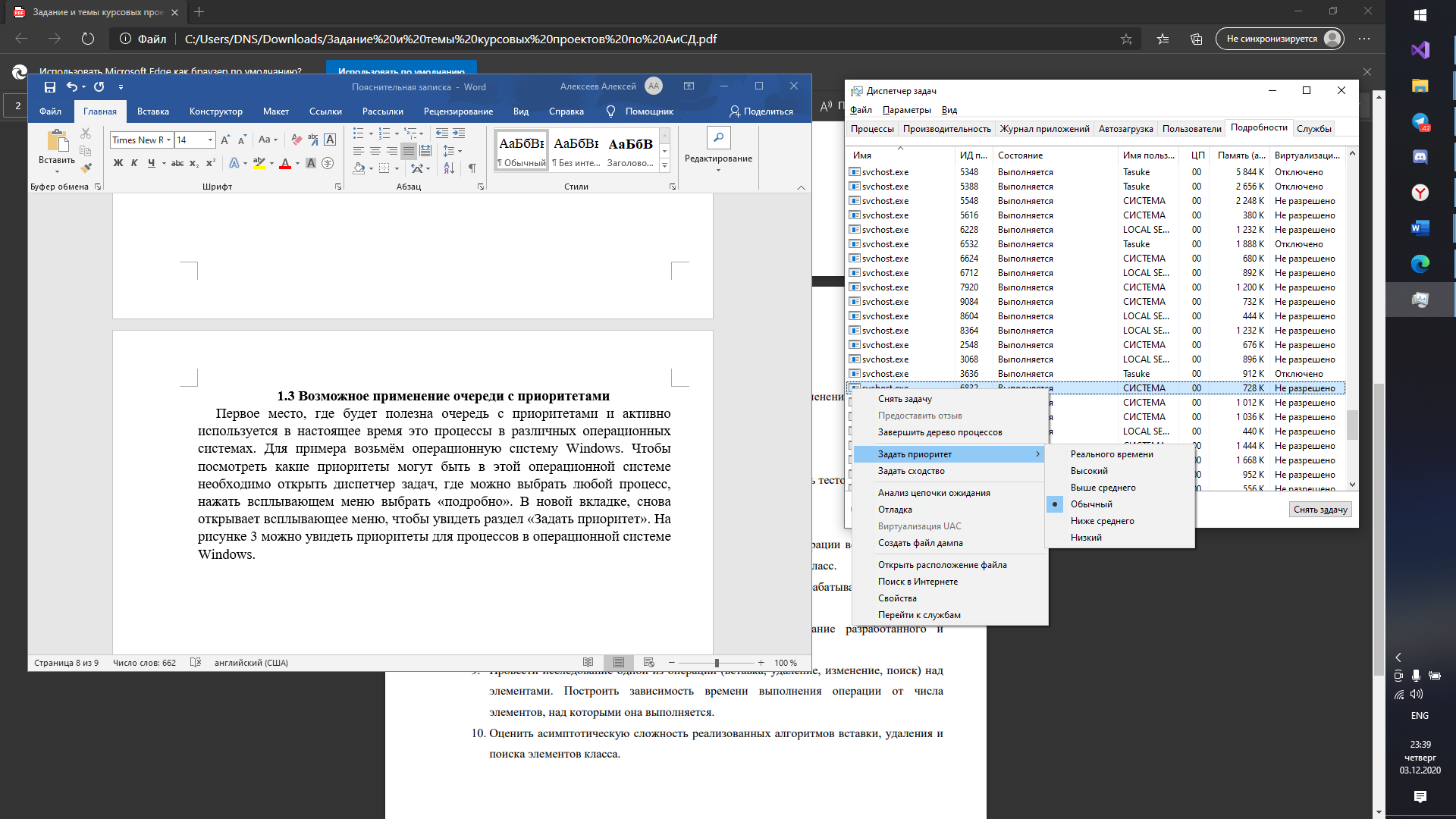


Рисунок 4 – Приоритеты процессов в Windows

Всего приоритетов в операционной системе Windows 6:

1. Реального времени
2. Высокий
3. Выше среднего
4. Обычный
5. Ниже среднего
6. Низкий

Приоритет под номер 1 самый высокий, а приоритет под номером 6 самый низкий.

# 2 Проектирование

На этапе проектирования необходимо обдумать предполагаемый алгоритм жизненного цикла программы, который удовлетворял бы всем требуемым задачам и был бы универсален, то есть, если поступят новые требования или появятся новые цели к реализации новых функций, необходимо, чтобы программу можно было дописать максимально безболезненно и с минимальными поправками в существующем коде. Для этого нужно на этом этапе выбрать наиболее универсальную и приспосабливаемую модель поведения программы.

## 2.1 Техническое задани

* Спроектировать структуру для очереди с приоритетом

Поля:

* Коллекция с элементами очереди (тип данных одна из стандартных коллекций в языке программирования С#)

Свойства:

* Пустая ли коллекция (тип данных: логический)

Методы:

* Добавление в очередь (тип данных: пустой)
* Получение элемента (первого) из очереди (тип данных: элемент очереди)
* Получение индексах элемента по объекту (тип данных: целочисленные числа)
* Проверка на наличие элемента с заданным ключом в очереди (тип данных: логический)
* Спроектировать структуру элемента очереди

Поля:

* Ключ (тип данных: целочисленные числа)
* Значение (тип данных: числа с плавающей запятой)
* Приоритет элемента (тип данных: перечисление с приоритетами)
* Спроектировать интерфейс тестового приложения

Возможности:

* Добавление элемента с заданным ключом, значением и приоритетом в очередь.
* Получение элемента из очереди.

Запуск тестирования времени добавления элементов для очереди с приоритетным добавлением и запуск тестирования времени получения элемента для очереди с приоритетным удалением.

## 2.2 Проектирование класса очереди

На рис. 6 можно увидеть описание класса Item, а на рис. 7 описание интерфейса для класса очереди.

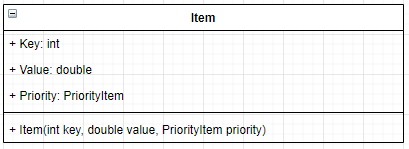


Рисунок 5 – Описание класса Item

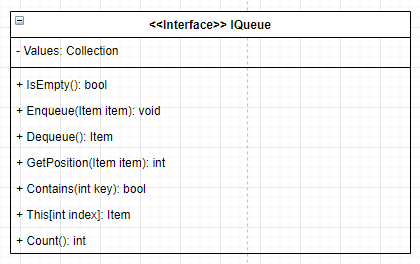


Рисунок 6 – Описание интерфейса для класса очереди

Представлено описание интерфейса для класса очереди. Принято решение не использовать наследование, чтобы избежать сильной связанности классов наследников от изменений в классе родителя, если они понадобятся.

Дальше (в таблице 1) можно прочитать описание всех полей и методов интерфейса IQueue.

Таблица 1 – Описание интерфейса IQueue

|  |  |
| --- | --- |
| Название | Назначение |
| Values | Коллекция элементов очереди, которая содержит в себе объекты типа IItem. Коллекция является приватный полем. |
| IsEmpty | Проверяется количество элементов в Values, если 0, то возвращается false, иначе true. В языке программирования C# |
| Enqueue | Метод добавления элемента в очередь. |
| Dequeue | Метод получения элемента из очереди. Возвращается первый элемент с самым высоким приоритетом их всех элементов |
| GetPosition | Метод возвращает индекс элемента по значению ключа. При отсутствии элемента с введённым ключом выбрасывается исключение, чтобы можно было отследить данную ситуацию |
| Contains | Метод проверяет присутствие элемента по введённому ключу. Если элемент с введённым ключом есть в очереди возвращается true, иначе false |
| This[index] | Индексирование для класса очереди. Индексирование осуществляется к коллекции значений (Values). С помощью индексирования можно как просто получать элементы из коллекции-хранилища, так и изменять. |
| Count | Метод, возвращающий количество элементов в очереди |

Необходимость сокрытия основной коллекции, где хранятся элементы очереди можно объяснить тем, что это предотвратит возможность прямого изменения элементов очереди. Производится изменения над элементами очереди должны только через доступные методы класса.

Тестовое приложение должно давать не только возможность проверить работу основных методов очереди, но и дать возможность провести ряд тестов для оценки времени удаления элементов из очереди. Если будет возможно, то сразу строить график по полученным данным, чтобы не тратить время на перенос данных в другие программы для построения графиков. Самое главное потребуется сохранять данные тестов в файл, чтобы их можно было добавить, как доказательство. Предполагаемый вид тестового приложения можно увидеть на рисунке 7.

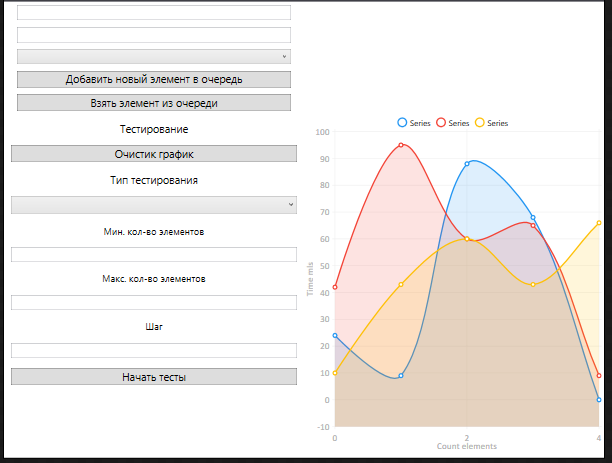


Рисунок 7 – Дизайн тестового приложения

Таким образом, на этапе проектирования рассмотрена вся структура будущего класса, спланировано поведение каждого метода и решено, какой формат будет тестового приложения. В процессе кодирования может появиться нужда во вспомогательных методах класса, которые могут понадобиться в рассмотренных, поэтому в нём могут добавиться ещё приватные методы.

# 3 Кодирование и отладка

## 3.1 Использование класса

Для создания и редактирования очереди с приоритетами необходим объект, который может быть создан двумя способами: с помощью конструктора, в который предаётся другая очередь с приоритетами, то есть конструктор копирования; во второй конструктор ничего не передаётся и создаётся пустая очередь.

## 3.2 Основные методы класса QueuePriorityDelete

Далее список методов с объяснением внутренней логики:

* QueuePriorityDelete():

Конструктор класса. Ничего не принимает, создаёт пустую очередь, также создаёт объект коллекция \_values, где хранятся все элементы очереди.

* QueuePriorityDelete(ObservableCollection<IItem> values):

Конструктор копирования. Принимает объект класса ObservableCollection, хранящий элементы типа IItem. В начале конструктора инициализируется \_values, после проходим по всем значения коллекции values и добавляем их копии в \_values.

* void Enqueue(IItem item):

Так как эта реализация очереди с приоритетным удалением, то элемент добавляется в конец \_values, вызовом у последнего метода Add().

* IItem Dequeue():

В методе изъятия элемента из очереди нужно учитывать приоритет элементов, по этой причине создаётся локальная переменная (highestPriority) в котором будет хранится самый высокий приоритет, на текущий момент, изначально присваивается самый низкий приоритет, если встречается элемент с приоритетом выше, то значение переменной меняется на приоритет этого элемента и сохраняется индекс этого элемента в переменную indexRemove. Если значение приоритета в переменной дойдёт до самого высокого, то сразу выходим из цикла, нет смысла просматривать остальные элементы. В локальную переменную сохраняем элемент, который будем удалять из очереди, после чего удаляем его из коллекции \_values, вызовом у него метода RemoveAt(indexRemove), и в конце возвращаем удалённый элемент.

* bool IsEmpty:

В терминологии языка программирования С#, IsEmpty – это свойство, которое имеет только геттер. Он возвращает true, если количество элементов меньше 1, иначе возвращает false.

* int GetPosition(IItem item):

Этот метод выбрасывает исключение, если ключ искомого элемента отсутствует в очереди. Если элемент с таким ключом всё же есть, то методом First у \_values, получаем первый элемент с необходимым ключом, а после чего с помощью метода IndexOf у\_values, получаем индекс этого элемента.

* bool Contains(int key):

Этот метод использует (напрямую возвращает значение метода Any) метод Any у \_values. Он работает так: по заданному условию возвращает логическое значение, есть ли хотя бы один элемент или нет. True – если есть хотя бы один элемент под заданное условие, иначе false.

* int Count:

Это свойство напрямую возвращает значение свойства Count у \_values.

* IItem this[int index]:

Реализация поддержки индексирование для объекта класса очереди. Состоит из двух частей сеттера и геттера, если сеттер не реализовывать, то присваивать новый объект элемента через индексирование будет нельзя. И геттер, и сеттер могут выбросить исключение, если индекс будет меньше 0 или больше количества элементов в \_values.

Класса не был помечен, как final, так что имеется возможность его наследования, и переопределение всех методов кроме: Enqueue и Dequeue. Для возможности переопределения методы были помечены ключевым словом virtual. Так же стоит отметить, что коллекция с элементами очереди отмечена ключевым словом readonly, то есть это поле можно инициализировать только внутри конструкторов класса, и нигде более.

# 4 Тестирование

­На этапе тестирования разработчик должен постараться, чтобы его программный продукт вёл себя адекватно при любых обстоятельствах – и при часто используемых действиях, и при частных случаях. Тестировать программный продукт необходимо в двух направлениях: в адекватности работы стандартных методов очереди с приоритетами и в адекватности при замерах времени при удалении или добавлении элемента и построении графика.

## 4.1 работоспособность очереди с приоритетом

Для удобства тестирования в окне тестового приложения выводится список элементов очереди с приоритетами, так как они хранятся внутри неё (в поле \_values). Это можно увидеть на рисунке 8.

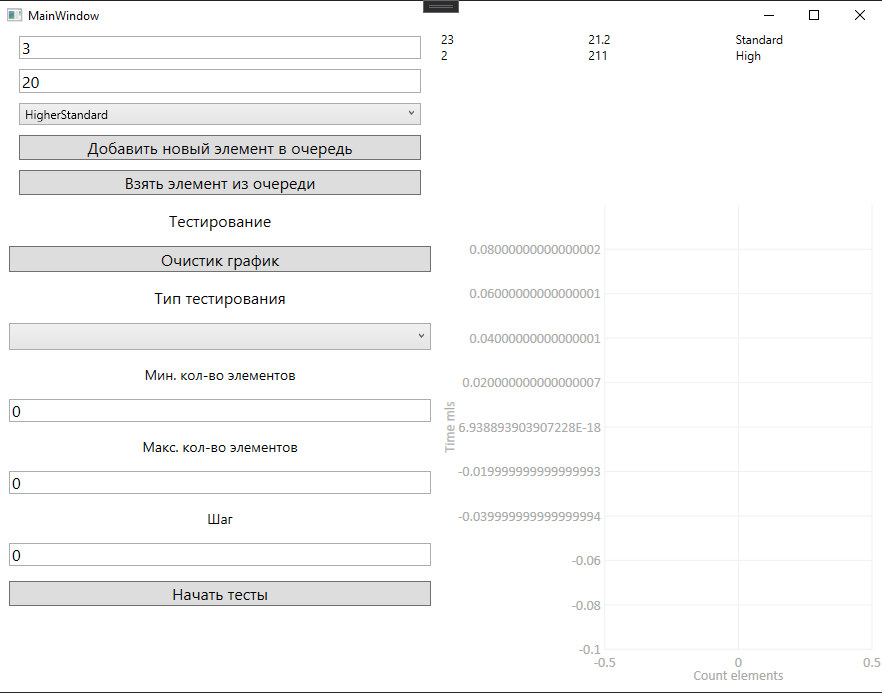


Рисунок 8 – Интерфейс реализованного приложения

После того, как мы нажмём на кнопку в очередь, добавится элемент с параметрами: ключ = 3, значение = 20, приоритет = выше среднего. Это можно увидеть на рисунке 9 в правой, верхней части приложения.

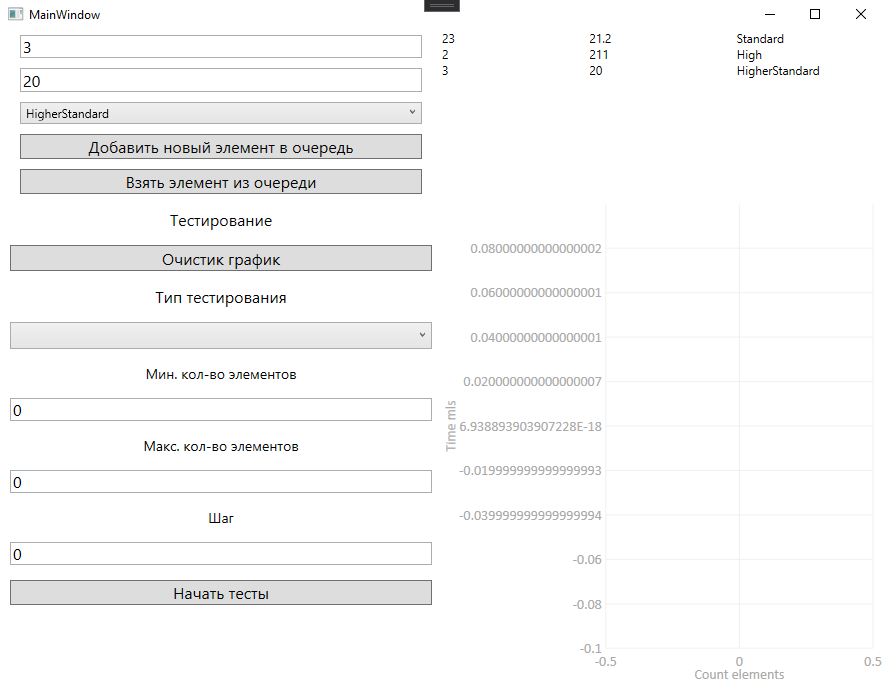


Рисунок 9 – Состояние приложения после добавления элемента

Можно заметить, что элемент добавился в конец очереди, это происходит потому, что была реализована очередь с приоритетным удалением, то есть мы увидим применение приоритетов, только при тестировании удаления. Результат удаления элемента из очереди можно увидеть на рисунке 10.

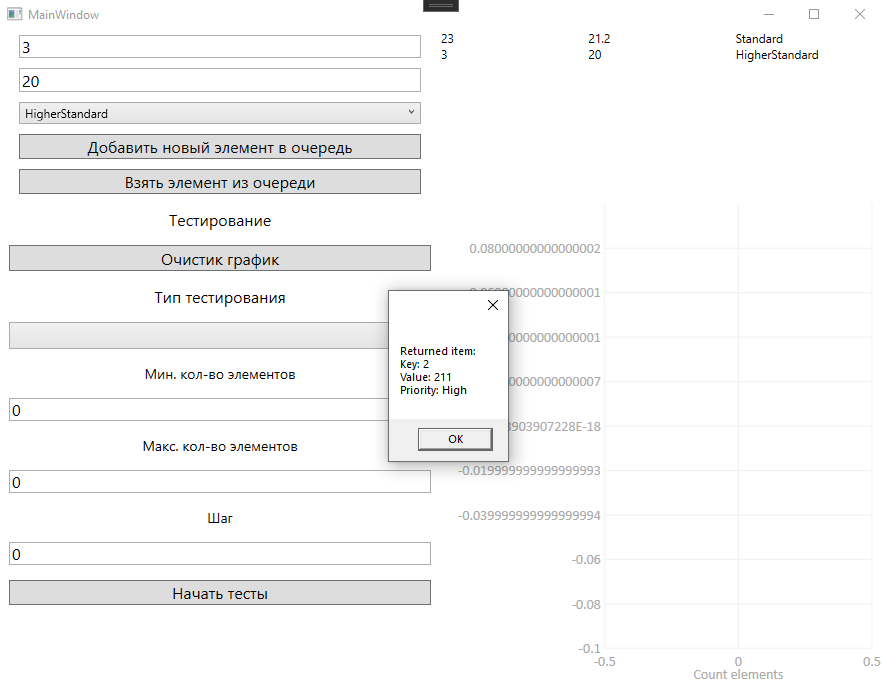


Рисунок 10 – Состояние приложения после удаления элемента

Важно отметить, что удаление произошло корректно, так как именно элемент с ключом 2 имел самый высокий приоритет (высокий), но при этом находился в центре всей коллекции элементов очереди.

## 4.2 Тестирование оценки времени

Убедившись, что добавление и удаление происходит корректно, можно начать тестирование оценки времени и построения графика по полученным данным. Первым делом протестируем проведение тестов вставки и удалении, и формирование файлов с выходными данными. Результат можно увидеть на рисунке 11-16.

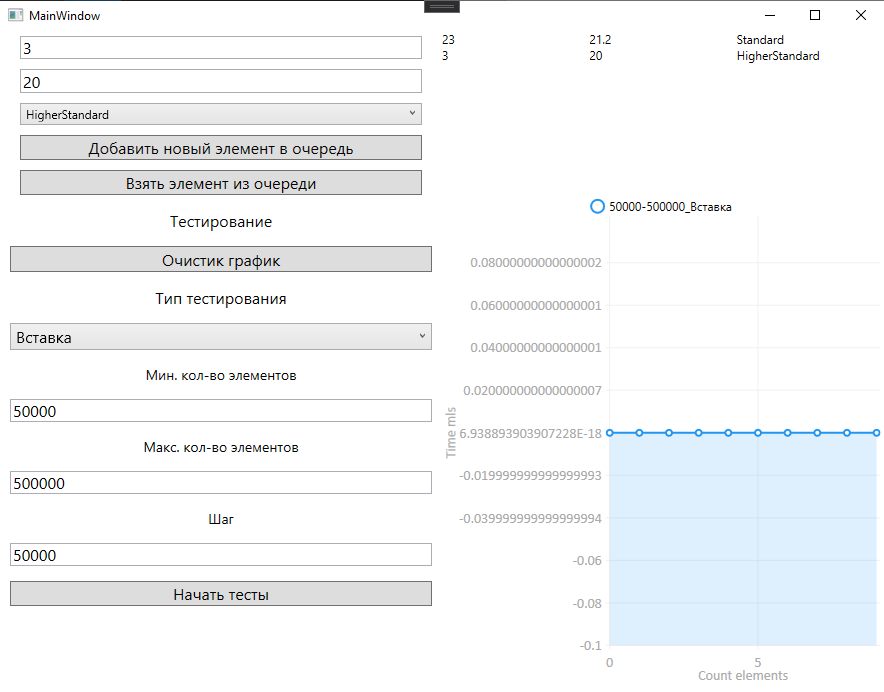


Рисунок 11 – Состояние приложения после проведение тестов на вставку элементов

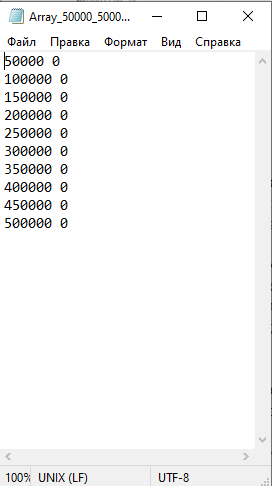


Рисунок 12 – Выходные данные тестов на вставку элементов

В файле можно увидеть два столбца со значениями, первый столбец это количество элементов в очереди. Второй столбец это время затраченное в милисекундах.

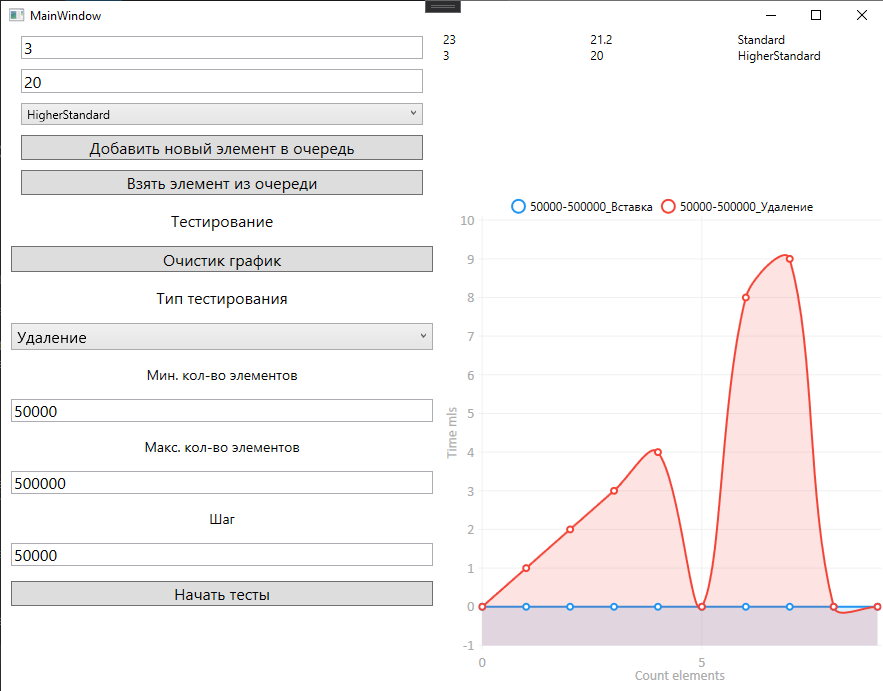


Рисунок 13 – Состояние приложения после тестов на удаление элементов из очереди

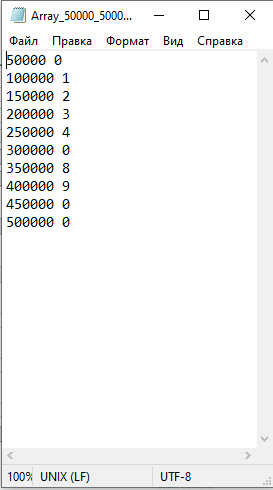


Рисунок 14 – Выходные данные тестов на удаление элементов

По текущим данным, которые находятся в приложении Б, таблица 2, невозможно судить о какой-либо зависимости времени от количества элементов. Чтобы получить больше данных для анализа стоит увеличить максимальное количество элементов очереди до 5 миллионов элементов, тогда будет 100 точек для построения графика.

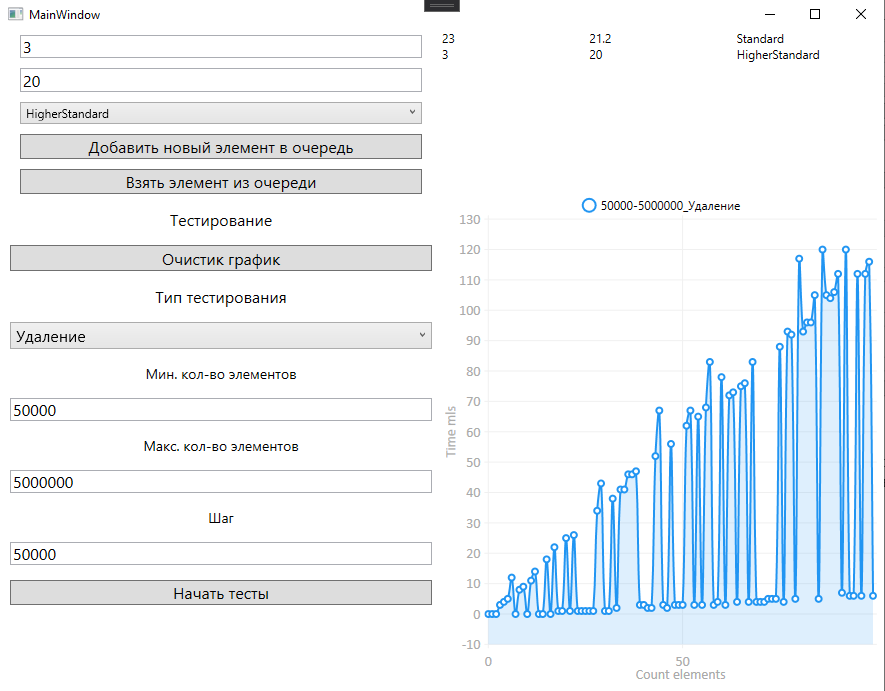


Рисунок 15 – Состояние приложения после тестов на удаление элементов

По полученному графику можно предположить, что зависимость в среднем случаем всё же есть, а зависимость эта линейная. Данные к графику находятся в приложении Б, таблица 3. Увеличим максимальное количество элементов до 50 миллионов и проведём ещё ряд тестов, чтобы убедиться в этом.

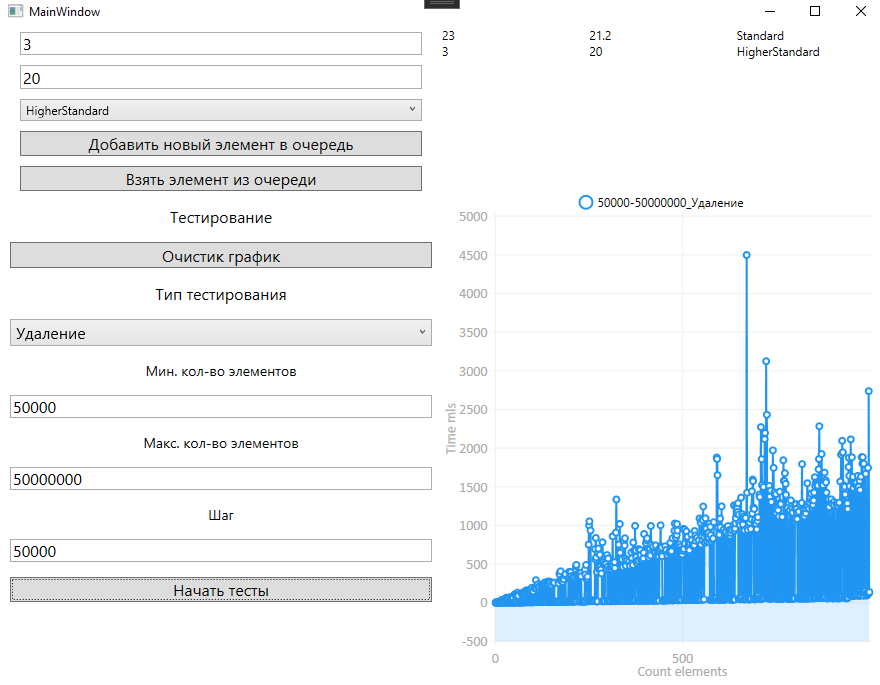


Рисунок 16 – Состояние приложения после тестов на удаление элемента

На графике можно проследить тренд на планомерное увеличение времени при удалении элемента при росте количества элементов в очереди. Однако, как и ранее, можно увидеть, что есть лучший случай, когда затрачивается близкое к нулю время на удаление элемента. И есть худший случай, когда затрачивается сильно больше времени относительно других тестов.

Для лучшей наглядности график для последних тестов был построен ещё в программе Excel (рисунок 17), так же на графике проведена линия тренда (красная линия), а данные, по которым был построен график можно увидеть в приложении Б, таблица 4.

Рисунок 17 – График зависимости времени удаления от количества элементов

## 4.3 Оценка асимптотической сложности реализованных алгоритмов

Оценку асимптотической сложности проводим на код в этих методах.

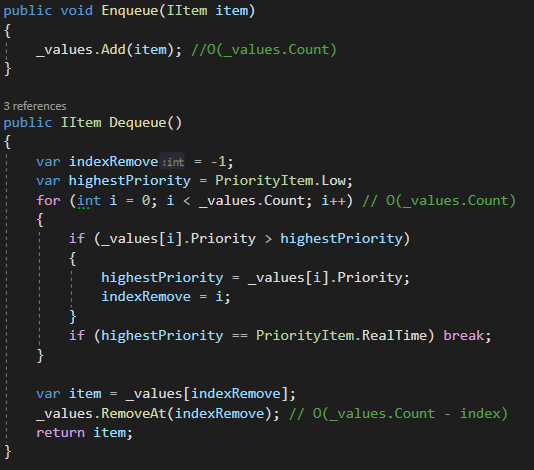


Рисунок 18 – Реализация вставки и удаления

Получаем, что асимптотическая сложность:

* Вставки О(N), так как добавление происходит всегда в конец.
* Удаления О(2N), так как сперва проходим по массиву и ищем элемент для удаления, потом выполняем само удаление.

# 5 Сопровождение

Программный продукт компилировался с помощью .Net 5 в операционной системе Windows 10, языковой стандарт C# 9. Интерфейс программы представлен в виде оконного приложения, написанного на технологии WPF.

Интерфейс условно разделён на две части: левая половина окна, где вводятся данные и расположены кнопки для воспроизведения действий; и правая часть окна, где находятся визуальные объекты для отображения результата. Сверху справа список с элементами очереди, которые добавляются вручную, ниже находятся график для наглядного отображения проведённых тестов.

При нажатии на кнопку «Добавить новый элемент в очередь» из первых двух полей берутся значения, ключ и значение соответственно, из combox’а берётся значение приоритета, если приоритет не указан, то выставляется автоматическое значение (Standard). После чего элемент появляется в списке на правой части приложения.

При нажатии на кнопку «Взять элемент из очереди» берётся первый элемент с самым высоким приоритетом. Появляется окно с данными элемента, который был изъят из очереди.

При нажатии на кнопку «Очистить график» удаляются все ранее построенные линии на графике.

При нажатии на кнопку «Начать тесты» из полей берутся значения для минимального и максимального количества элементов, а также шаг для изменения текущего количества элементов.

Результаты тестов сохраняются в текстовый файл с форматом имени «Array\_ <мин. кол-во элементов> \_ <макс. кол-во элементов> \_<шаг>\_1.txt». Данные из этих файлов можно использовать для построения графиков в любой другой программе.

# Заключение

В ходе выполнения курсового проекта были на практике закреплены знания о языке программирования С#; Реализован класс очереди с приоритетным удалением; Спроектировано и реализовано тестовое приложение; Проведены ряд тестов для оценки времени удаления элемента от количества элементов в очереди. Эти тесты показали, что зависимость есть, и она линейная.

Поставленная цель выполнена, все задачи выполнены.

# Список использованной литературы

1. Пример реализации очереди с приоритетами для шаблонного класса. реализация в виде динамического массива // https://www.bestprog.net/ru/2019/09/29/c-an-example-implementation-of-a-priority-queue-for-a-template-class-implementation-as-a-dynamic-array-ru/.

2. Очередь с приоритетом C++, C# (си шарп) // https://wiki.dieg.info/ochered\_s\_prioritetom.

3. Шаблон очереди с приоритетом // https://docs.microsoft.com/ru-ru/azure/architecture/patterns/priority-queue.

4. Очередь с приоритетом (программирование) // https://ru.wikipedia.org/wiki/Очередь\_с\_приоритетом\_(программирование)

# Приложения

## Приложение А. Полные листинги файлов

### А.1 Файл IItem.cs

1. namespace PriorityQueue
2. {
3. public interface IItem
4. {
5. int Key { get; set; }
6. double Value { get; set; }
7. public PriorityItem Priority { get; set; }
8. }
9. }

### А.2 Файл Item.cs

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. namespace PriorityQueue
7. {
8. public enum PriorityItem
9. {
10. Low,
11. LowerStandard,
12. Standard,
13. HigherStandard,
14. High,
15. RealTime,
16. }
17. public class Item : IItem
18. {
19. public Item(int key, double value, PriorityItem priority = PriorityItem.Standard)
20. {
21. Key = key;
22. Priority = priority;
23. Value = value;
24. }
25. public int Key { get; set; }
26. public PriorityItem Priority { get; set; }
27. public double Value { get; set; }
28. }
29. }

### А.3 Файл IQueue.CS

1. namespace PriorityQueue
2. {
3. public interface IQueue
4. {
5. void Enqueue(IItem item);
6. IItem Dequeue();
7. bool IsEmpty { get; }
8. int GetPosition(IItem item);
9. bool Contains(int key);
10. int Count { get; }
11. IItem[] ToArray();
12. }
13. }

### А.4 Файл QueuePriorityDelete.cs

1. using System;
2. using System.Collections.Generic;
3. using System.Collections.ObjectModel;
4. using System.Linq;
5. using System.Security.Cryptography.X509Certificates;
6. using System.Text;
7. using System.Threading.Tasks;
8. namespace PriorityQueue
9. {
10. public class QueuePriorityDelete : IQueue
11. {
12. private readonly ObservableCollection<IItem> \_values;
13. public QueuePriorityDelete()
14. {
15. \_values = new ObservableCollection<IItem>();
16. }
17. public QueuePriorityDelete(QueuePriorityDelete queue)
18. {
19. \_values = new ObservableCollection<IItem>();
20. for (int i = 0; i < queue.Count; i++)
21. {
22. Enqueue(new Item(queue[i].Key, queue[i].Value, queue[i].Priority));
23. }
24. }
25. public void Enqueue(IItem item)
26. {
27. \_values.Add(item);
28. }
29. public IItem Dequeue()
30. {
31. var indexRemove = -1;
32. var highestPriority = PriorityItem.Low;
33. for (int i = 0; i < \_values.Count; i++)
34. {
35. if (\_values[i].Priority > highestPriority)
36. {
37. highestPriority = \_values[i].Priority;
38. indexRemove = i;
39. }
40. if (highestPriority == PriorityItem.RealTime) break;
41. }
42. var item = \_values[indexRemove];
43. \_values.RemoveAt(indexRemove);
44. return item;
45. }
46. public virtual bool IsEmpty => \_values.Count < 1;
47. public virtual int GetPosition(IItem item)
48. {
49. if (this.Contains(item.Key) == false)
50. throw new Exception("Элемента с таким ключом нет в очереди");
51. return \_values.IndexOf(\_values.First(x => x.Key == item.Key));
52. }
53. public virtual bool Contains(int key)
54. {
55. return \_values.Any(x => x.Key == key);
56. }
57. public virtual int Count => \_values.Count;
58. public IItem[] ToArray()
59. {
60. return \_values.ToArray();
61. }
62. public virtual IItem this[int index]
63. {
64. get
65. {
66. if (index < 0 || index >= \_values.Count)
67. throw new Exception("Incorrect index.\n");
68. return \_values[index];
69. }
70. set
71. {
72. if (index < 0 || index >= \_values.Count)
73. throw new Exception("Incorrect index.\n");
74. \_values[index] = value;
75. }
76. }
77. }
78. }

### А.5 Файл BaseRelayCommand.CS

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. using System.Windows.Input;
7. namespace Calc.Commands
8. {
9. public abstract class BaseRelayCommand : ICommand
10. {
11. public abstract void Execute(object parameter);
12. public abstract bool CanExecute(object parameter);
13. public event EventHandler CanExecuteChanged
14. {
15. add { CommandManager.RequerySuggested += value; }
16. remove { CommandManager.RequerySuggested -= value; }
17. }
18. }
19. }

### А.5 Файл RelayCommand.cs

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. namespace Calc.Commands
7. {
8. public class RelayCommand<T> : BaseRelayCommand
9. {
10. private readonly Action<T> \_execute;
11. private readonly Func<T, bool> \_canExecute;
12. public RelayCommand(Action<T> execute, Func<T, bool> canExecute = null)
13. {
14. \_execute = execute ?? throw new ArgumentNullException(nameof(execute));
15. \_canExecute = canExecute;
16. }
17. public override void Execute(object parameter)
18. {
19. if (parameter is T arg)
20. {
21. \_execute.Invoke(arg);
22. }
23. }
24. public override bool CanExecute(object parameter)
25. {
26. if (parameter is T arg)
27. {
28. return \_canExecute?.Invoke(arg) ?? true;
29. }
30. return false;
31. }
32. }
33. public class RelayCommand : BaseRelayCommand
34. {
35. private readonly Action \_execute;
36. private readonly Func<bool> \_canExecute;
37. public RelayCommand(Action execute, Func<bool> canExecute = null)
38. {
39. \_execute = execute ?? throw new ArgumentNullException(nameof(execute));
40. \_canExecute = canExecute;
41. }
42. public override void Execute(object parameter)
43. {
44. \_execute.Invoke();
45. }
46. public override bool CanExecute(object parameter)
47. {
48. return \_canExecute?.Invoke() ?? true;
49. }
50. }
51. }

### А.6 Файл MainViewModel.cs

1. using System;
2. using System.Collections.Generic;
3. using System.Collections.ObjectModel;
4. using System.ComponentModel;
5. using System.IO;
6. using System.Linq;
7. using System.Runtime.CompilerServices;
8. using System.Text;
9. using System.Threading;
10. using System.Threading.Tasks;
11. using System.Windows;
12. using System.Windows.Input;
13. using Calc.Commands;
14. using LiveCharts;
15. using LiveCharts.Wpf;
16. using PriorityQueue.Annotations;
17. namespace PriorityQueue
18. {
19. class MainViewModel : INotifyPropertyChanged
20. {
21. public long MinElements { get; set; }
22. public long Step { get; set; }
23. public long MaxElements { get; set; }
24. public int[] LastResults { get; set; }
25. public IItem[] QueueItems => Queue.ToArray();
26. public SeriesCollection SeriesCollection { get; set; }
27. private void UpdateSeriesCollection(LineSeries line)
28. {
29. SeriesCollection.Add(line);
30. }
31. public string TypeTest(int type)
32. {
33. return type == 0 ? "Вставка" : "Удаление";
34. }
35. public ObservableCollection<Thread> ThreadCollection { get; set; }
36. public MainViewModel()
37. {
38. Queue = new QueuePriorityDelete();
39. Queue.Enqueue(new Item(23, 21.2, PriorityItem.Standard));
40. Queue.Enqueue(new Item(2, 211, PriorityItem.High));
41. ThreadCollection = new ObservableCollection<Thread>();
42. SeriesCollection = new SeriesCollection();
43. MinElements = 0;
44. }
45. /// <summary>
46. /// Do tests inserting or deleting
47. /// </summary>
48. /// <param name="typeTest">Inserting test = 0, Deleting test = 1</param>
49. public void DoTest(int typeTest, int typeQueue = 0)
50. {
51. IQueue queue = new QueuePriorityDelete();
52. int indexTest = 0;
53. Random random = new Random(DateTime.Now.Millisecond);
54. string message = "";
55. LastResults = new int[(MaxElements - MinElements) / Step + 1];
56. for (long j = MinElements; j <= MaxElements; j += Step, indexTest++)
57. {
58. for (long i = queue.Count; i < j; i++)
59. {
60. queue.Enqueue(new Item((int)i, 0.2 + i, (PriorityItem)random.Next(0,7)));
61. }
62. //Check time
63. DateTime startTime;
64. DateTime endTime;
65. if (typeTest == 0)
66. {
67. startTime = DateTime.Now;
68. queue.Enqueue(new Item(queue.Count + 5, random.NextDouble(), (PriorityItem)random.Next(0, 7)));
69. endTime = DateTime.Now;
70. }
71. else
72. {
73. startTime = DateTime.Now;
74. queue.Dequeue();
75. endTime = DateTime.Now;
76. }
78. var time = LastResults[indexTest] = (int)(endTime - startTime).TotalMilliseconds;
79. message += $"{j} {time}\n";
80. }
81. File.WriteAllText($"Array\_{MinElements}\_{MaxElements}\_{Step}\_{typeTest}.txt", message);
82. //Create new line in chart
83. SeriesCollection.Add(new LineSeries
84. {
85. Title = $"{MinElements}-{MaxElements}\_{TypeTest(typeTest)}",
86. Values = new ChartValues<int>(LastResults)
87. });
88. }
89. private IQueue \_queue;
90. public IQueue Queue
91. {
92. get => \_queue;
93. set
94. {
95. \_queue = value;
96. OnPropertyChanged(nameof(Queue));
97. }
98. }
99. public int Key { get; set; }
100. public double ValueItem { get; set; }
101. public PriorityItem Priority { get; set; }
102. public ICommand AddElementInQueue => new RelayCommand((() =>
103. {
104. Queue.Enqueue(new Item(Key, ValueItem, Priority));
105. OnPropertyChanged(nameof(QueueItems));
106. }), () => true);
107. public ICommand DeleteElementFromQueue => new RelayCommand((() =>
108. {
109. var item = Queue.Dequeue();
110. OnPropertyChanged(nameof(QueueItems));
111. MessageBox.Show($"Returned item:\nKey: {item.Key}\nValue: {item.Value}\nPriority: {item.Priority}");
112. }), () => Queue.Count > 0);
113. public event PropertyChangedEventHandler? PropertyChanged;
114. [NotifyPropertyChangedInvocator]
115. protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
116. {
117. PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
118. }
119. }
120. }

### А.7 Файл MainWindow.xaml

1. <Window x:Class="PriorityQueue.MainWindow"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6. xmlns:local="clr-namespace:PriorityQueue"
7. xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
8. mc:Ignorable="d"
9. Title="MainWindow" Height="750" Width="900"
10. WindowStartupLocation="CenterScreen"
11. d:DataContext="{d:DesignInstance local:MainViewModel}">
12. <Grid>
13. <Grid.ColumnDefinitions>
14. <ColumnDefinition/>
15. <ColumnDefinition/>
16. </Grid.ColumnDefinitions>
17. <StackPanel Grid.Column="0"
18. Orientation="Vertical"
19. >
20. <Label Content="Ключ"
21. HorizontalContentAlignment="Center"
22. FontSize="16" Margin="5"></Label>
23. <TextBox Text="{Binding Key}"
24. Margin="20 5"
25. FontSize="16"></TextBox>
26. <Label Content="Значение элемента"
27. HorizontalContentAlignment="Center"
28. FontSize="16" Margin="5"></Label>
29. <TextBox Text="{Binding ValueItem}"
30. Margin="20 5"
31. FontSize="16"></TextBox>
32. <Label Content="Приоритет"
33. HorizontalContentAlignment="Center"
34. FontSize="16" Margin="5"></Label>
35. <ComboBox Name="BoxPriority"
36. Margin="20 5"
37. Text="Priority" SelectedIndex="{Binding Priority}">
38. <TextBlock Text="Low"></TextBlock>
39. <TextBlock Text="LowerStandard"></TextBlock>
40. <TextBlock Text="Standard"></TextBlock>
41. <TextBlock Text="HigherStandard"></TextBlock>
42. <TextBlock Text="High"></TextBlock>
43. <TextBlock Text="RealTime"></TextBlock>
44. </ComboBox>
45. <Button Content="Добавить новый элемент в очередь"
46. Command="{Binding AddElementInQueue}"
47. FontSize="16"
48. Margin="20 5"></Button>
49. <Button Content="Взять элемент из очереди"
50. FontSize="16"
51. Command="{Binding DeleteElementFromQueue}"
52. Margin="20 5"></Button>
53. <Label Content="Тестирование"
54. HorizontalContentAlignment="Center"
55. FontSize="16" Margin="5"></Label>
56. <Button Content="Очистик график"
57. Click="ButtonClick\_ClearChart"
58. FontSize="16"
59. Margin="10 5"></Button>
60. <Label Content="Тип тестирования"
61. FontSize="16"
62. Margin="10 5"
63. HorizontalContentAlignment="Center"/>
64. <ComboBox FontSize="16"
65. Margin="10 5"
66. Name="ComboBoxTypeTest">
67. <TextBlock Text="Вставка"></TextBlock>
68. <TextBlock Text="Удаление"></TextBlock>
69. </ComboBox>
70. <Label Content="Мин. кол-во элементов"
71. HorizontalContentAlignment="Center"
72. FontSize="14" Margin="10 5"></Label>
73. <TextBox FontSize="16"
74. Margin="10 5"
75. Text="{Binding MinElements}"></TextBox>
76. <Label Content="Макс. кол-во элементов"
77. HorizontalContentAlignment="Center"
78. FontSize="14" Margin="10 5"></Label>
79. <TextBox FontSize="16"
80. Margin="10 5"
81. Text="{Binding MaxElements}"></TextBox>
82. <Label Content="Шаг"
83. HorizontalContentAlignment="Center"
84. FontSize="14" Margin="10 5"></Label>
85. <TextBox FontSize="16"
86. Margin="10 5"
87. Text="{Binding Step}"></TextBox>
88. <Button Content="Начать тесты"
89. FontSize="16" Margin="10"
90. Click="ButtonClick\_DoTest"></Button>
91. </StackPanel>
92. <Grid Grid.Column="1">
93. <Grid.RowDefinitions>
94. <RowDefinition Height="\*"/>
95. <RowDefinition Height="3\*"/>
96. </Grid.RowDefinitions>
97. <ItemsControl Grid.Row="0"
98. ItemsSource="{Binding QueueItems,
99. Mode=OneWay, UpdateSourceTrigger=PropertyChanged}">
100. <ItemsControl.ItemTemplate>
101. <DataTemplate>
102. <Grid>
103. <Grid.ColumnDefinitions>
104. <ColumnDefinition/>
105. <ColumnDefinition/>
106. <ColumnDefinition/>
107. </Grid.ColumnDefinitions>
108. <TextBlock Grid.Column="0" Text="{Binding Key,
109. Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"></TextBlock>
110. <TextBlock Grid.Column="1" Text="{Binding Value,
111. Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"></TextBlock>
112. <TextBlock Grid.Column="2" Text="{Binding Priority,
113. Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"></TextBlock>
114. </Grid>
115. </DataTemplate>
116. </ItemsControl.ItemTemplate>
117. </ItemsControl>
118. <lvc:CartesianChart Grid.Row="1"
119. Series="{Binding SeriesCollection}"
120. LegendLocation="Top" >
121. <lvc:CartesianChart.AxisY>
122. <lvc:Axis Title="Time mls"
123. FontSize="14"
124. Labels="{Binding LastResults}"></lvc:Axis>
125. </lvc:CartesianChart.AxisY>
126. <lvc:CartesianChart.AxisX>
127. <lvc:Axis Title="Count elements"
128. FontSize="14"></lvc:Axis>
129. </lvc:CartesianChart.AxisX>
130. </lvc:CartesianChart>
131. </Grid>
132. </Grid>
133. </Window>

### А.8 Файл MainWindow.xaml.cs

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading;
6. using System.Threading.Tasks;
7. using System.Windows;
8. using System.Windows.Controls;
9. using System.Windows.Data;
10. using System.Windows.Documents;
11. using System.Windows.Input;
12. using System.Windows.Media;
13. using System.Windows.Media.Imaging;
14. using System.Windows.Navigation;
15. using System.Windows.Shapes;
16. namespace PriorityQueue
17. {
18. /// <summary>
19. /// Interaction logic for MainWindow.xaml
20. /// </summary>
21. public partial class MainWindow : Window
22. {
23. private MainViewModel \_viewModel;
24. public MainWindow()
25. {
26. DataContext = \_viewModel = new MainViewModel();
27. InitializeComponent();
28. }
29. private PriorityItem GetPriorityItem(string priority)
30. {
31. return priority switch
32. {
33. "Low" => PriorityItem.Low,
34. "LowerStandard" => PriorityItem.LowerStandard,
35. "Standard" => PriorityItem.Standard,
36. "HigherStandard" => PriorityItem.HigherStandard,
37. "High" => PriorityItem.High,
38. \_ => PriorityItem.Standard
39. };
40. }
41. private void ButtonClick\_DoTest(object sender, RoutedEventArgs e)
42. {
43. \_viewModel.DoTest(ComboBoxTypeTest.SelectedIndex,1);
44. MessageBox.Show("Finished");
45. }
46. private void ButtonClick\_ClearChart(object sender, RoutedEventArgs e)
47. {
48. \_viewModel.SeriesCollection.Clear();
49. }
50. }
51. }

## Приложение Б. Таблицы с данными тестов

### Б.1 Результаты тестов удаления с 50000 до 500000 с шагом 50000

Таблица 2 – Результаты тестов удаления для 50000-500000 элементов

|  |  |
| --- | --- |
| Количество элементов | Время в миллисекундах |
| 50000 | 0 |
| 100000 | 1 |
| 150000 | 2 |
| 200000 | 3 |
| 250000 | 4 |
| 300000 | 0 |
| 350000 | 8 |
| 400000 | 9 |
| 450000 | 0 |
| 500000 | 0 |

### Б.2 Результаты тестов удаления с 50000 до 5000000 с шагом 50000

Таблица 3 – Результаты тестов удаления 50000-5000000 элементов

|  |  |
| --- | --- |
| Количество элементов | Время в миллисекундах |
| 50000 | 0 |
| 100000 | 0 |
| 150000 | 0 |
| 200000 | 3 |
| 250000 | 4 |
| 300000 | 5 |
| 350000 | 12 |
| 400000 | 0 |
| 450000 | 8 |
| 500000 | 9 |
| 550000 | 0 |
| 600000 | 11 |
| 650000 | 14 |
| 700000 | 0 |
| 750000 | 0 |
| 800000 | 18 |
| 850000 | 0 |
| 900000 | 22 |
| 950000 | 1 |
| 1000000 | 1 |
| 1050000 | 25 |
| 1100000 | 1 |
| 1150000 | 26 |
| 1200000 | 1 |
| 1250000 | 1 |
| 1300000 | 1 |
| 1350000 | 1 |
| 1400000 | 1 |
| 1450000 | 34 |
| 1500000 | 43 |
| 1550000 | 1 |
| 1600000 | 1 |
| 1650000 | 38 |
| 1700000 | 2 |
| 1750000 | 41 |
| 1800000 | 41 |
| 1850000 | 46 |
| 1900000 | 46 |
| 1950000 | 47 |
| 2000000 | 3 |
| 2050000 | 3 |
| 2100000 | 2 |
| 2150000 | 2 |
| 2200000 | 52 |
| 2250000 | 67 |
| 2300000 | 3 |
| 2350000 | 2 |
| 2400000 | 56 |
| 2450000 | 3 |
| 2500000 | 3 |
| 2550000 | 3 |
| 2600000 | 62 |
| 2650000 | 67 |
| 2700000 | 3 |
| 2750000 | 65 |
| 2800000 | 3 |
| 2850000 | 68 |
| 2900000 | 83 |
| 2950000 | 3 |
| 3000000 | 4 |
| 3050000 | 78 |
| 3100000 | 3 |
| 3150000 | 72 |
| 3200000 | 73 |
| 3250000 | 4 |
| 3300000 | 75 |
| 3350000 | 76 |
| 3400000 | 4 |
| 3450000 | 83 |
| 3500000 | 4 |
| 3550000 | 4 |
| 3600000 | 4 |
| 3650000 | 5 |
| 3700000 | 5 |
| 3750000 | 5 |
| 3800000 | 88 |
| 3850000 | 4 |
| 3900000 | 93 |
| 3950000 | 92 |
| 4000000 | 5 |
| 4 050 000 | 117 |
| 4100000 | 93 |
| 4150000 | 96 |
| 4200000 | 96 |
| 4 250 000 | 105 |
| 4300000 | 5 |
| 4 350 000 | 120 |
| 4 400 000 | 105 |
| 4 450 000 | 104 |
| 4 500 000 | 106 |
| 4 550 000 | 112 |
| 4600000 | 7 |
| 4 650 000 | 120 |
| 4700000 | 6 |
| 4750000 | 6 |
| 4 800 000 | 112 |
| 4850000 | 6 |
| 4 900 000 | 112 |
| 4 950 000 | 116 |
| 5000000 | 6 |

### Б.3 Результаты тестов удаления с 50000 до 50000000 с шагом 50000

Таблица 4 – Результаты тестов удаления с 50000-50000000 элементов

|  |  |
| --- | --- |
| 50000 | 1 |
| 100000 | 2 |
| 150000 | 5 |
| 200000 | 0 |
| 250000 | 0 |
| 300000 | 7 |
| 350000 | 8 |
| 400000 | 0 |
| 450000 | 11 |
| 500000 | 0 |
| 550000 | 0 |
| 600000 | 0 |
| 650000 | 0 |
| 700000 | 0 |
| 750000 | 1 |
| 800000 | 22 |
| 850000 | 33 |
| 900000 | 33 |
| 950000 | 1 |
| 1000000 | 38 |
| 1050000 | 1 |
| 1100000 | 29 |
| 1150000 | 30 |
| 1200000 | 31 |
| 1250000 | 1 |
| 1300000 | 46 |
| 1350000 | 47 |
| 1400000 | 43 |
| 1450000 | 1 |
| 1500000 | 65 |
| 1550000 | 2 |
| 1600000 | 55 |
| 1650000 | 45 |
| 1700000 | 2 |
| 1750000 | 56 |
| 1800000 | 2 |
| 1850000 | 2 |
| 1900000 | 2 |
| 1950000 | 2 |
| 2000000 | 73 |
| 2050000 | 3 |
| 2100000 | 54 |
| 2150000 | 55 |
| 2200000 | 3 |
| 2250000 | 2 |
| 2300000 | 2 |
| 2350000 | 62 |
| 2400000 | 64 |
| 2450000 | 3 |
| 2500000 | 3 |
| 2550000 | 4 |
| 2 600 000 | 109 |
| 2650000 | 3 |
| 2 700 000 | 109 |
| 2750000 | 4 |
| 2800000 | 91 |
| 2850000 | 4 |
| 2900000 | 4 |
| 2950000 | 92 |
| 3 000 000 | 131 |
| 3 050 000 | 104 |
| 3100000 | 91 |
| 3150000 | 4 |
| 3200000 | 4 |
| 3250000 | 4 |
| 3300000 | 4 |
| 3350000 | 4 |
| 3400000 | 4 |
| 3450000 | 4 |
| 3500000 | 5 |
| 3 550 000 | 115 |
| 3 600 000 | 118 |
| 3650000 | 4 |
| 3700000 | 5 |
| 3 750 000 | 134 |
| 3 800 000 | 129 |
| 3850000 | 5 |
| 3 900 000 | 112 |
| 3950000 | 5 |
| 4 000 000 | 118 |
| 4050000 | 5 |
| 4100000 | 6 |
| 4 150 000 | 161 |
| 4 200 000 | 133 |
| 4250000 | 5 |
| 4 300 000 | 157 |
| 4 350 000 | 139 |
| 4 400 000 | 145 |
| 4450000 | 6 |
| 4 500 000 | 148 |
| 4 550 000 | 152 |
| 4600000 | 6 |
| 4 650 000 | 155 |
| 4700000 | 6 |
| 4 750 000 | 177 |
| 4800000 | 6 |
| 4850000 | 9 |
| 4900000 | 6 |
| 4 950 000 | 194 |
| 5000000 | 8 |
| 5050000 | 7 |
| 5 100 000 | 147 |
| 5150000 | 7 |
| 5200000 | 7 |
| 5 250 000 | 190 |
| 5 300 000 | 197 |
| 5350000 | 7 |
| 5 400 000 | 189 |
| 5 450 000 | 292 |
| 5 500 000 | 180 |
| 5550000 | 9 |
| 5600000 | 9 |
| 5650000 | 9 |
| 5 700 000 | 238 |
| 5750000 | 8 |
| 5 800 000 | 188 |
| 5850000 | 9 |
| 5 900 000 | 212 |
| 5950000 | 8 |
| 6000000 | 8 |
| 6050000 | 9 |
| 6100000 | 8 |
| 6 150 000 | 234 |
| 6200000 | 9 |
| 6 250 000 | 217 |
| 6300000 | 8 |
| 6 350 000 | 206 |
| 6 400 000 | 242 |
| 6 450 000 | 229 |
| 6500000 | 9 |
| 6 550 000 | 217 |
| 6 600 000 | 260 |
| 6 650 000 | 221 |
| 6700000 | 9 |
| 6750000 | 10 |
| 6 800 000 | 241 |
| 6 850 000 | 258 |
| 6900000 | 10 |
| 6 950 000 | 269 |
| 7 000 000 | 265 |
| 7 050 000 | 247 |
| 7 100 000 | 274 |
| 7150000 | 11 |
| 7200000 | 11 |
| 7250000 | 10 |
| 7 300 000 | 274 |
| 7350000 | 10 |
| 7 400 000 | 260 |
| 7450000 | 10 |
| 7500000 | 10 |
| 7 550 000 | 228 |
| 7600000 | 12 |
| 7650000 | 10 |
| 7700000 | 10 |
| 7 750 000 | 257 |
| 7 800 000 | 258 |
| 7850000 | 11 |
| 7 900 000 | 259 |
| 7950000 | 11 |
| 8000000 | 11 |
| 8050000 | 13 |
| 8100000 | 13 |
| 8150000 | 12 |
| 8200000 | 13 |
| 8250000 | 13 |
| 8300000 | 12 |
| 8350000 | 11 |
| 8400000 | 12 |
| 8 450 000 | 271 |
| 8 500 000 | 251 |
| 8550000 | 13 |
| 8 600 000 | 380 |
| 8 650 000 | 364 |
| 8700000 | 12 |
| 8 750 000 | 407 |
| 8 800 000 | 284 |
| 8 850 000 | 283 |
| 8 900 000 | 293 |
| 8950000 | 12 |
| 9 000 000 | 307 |
| 9 050 000 | 270 |
| 9100000 | 13 |
| 9150000 | 12 |
| 9 200 000 | 303 |
| 9250000 | 13 |
| 9300000 | 12 |
| 9 350 000 | 353 |
| 9 400 000 | 370 |
| 9450000 | 14 |
| 9500000 | 13 |
| 9550000 | 14 |
| 9600000 | 14 |
| 9650000 | 16 |
| 9700000 | 14 |
| 9750000 | 15 |
| 9 800 000 | 316 |
| 9 850 000 | 321 |
| 9 900 000 | 400 |
| 9950000 | 13 |
| 10000000 | 16 |
| 10 050 000 | 309 |
| 10 100 000 | 395 |
| 10 150 000 | 325 |
| 10 200 000 | 345 |
| 10250000 | 16 |
| 10300000 | 14 |
| 10 350 000 | 304 |
| 10 400 000 | 320 |
| 10450000 | 15 |
| 10500000 | 14 |
| 10 550 000 | 348 |
| 10 600 000 | 334 |
| 10650000 | 16 |
| 10700000 | 19 |
| 10 750 000 | 431 |
| 10 800 000 | 472 |
| 10 850 000 | 490 |
| 10 900 000 | 344 |
| 10950000 | 18 |
| 11 000 000 | 366 |
| 11 050 000 | 322 |
| 11 100 000 | 336 |
| 11 150 000 | 353 |
| 11 200 000 | 383 |
| 11250000 | 17 |
| 11300000 | 16 |
| 11 350 000 | 342 |
| 11 400 000 | 383 |
| 11 450 000 | 340 |
| 11 500 000 | 342 |
| 11550000 | 15 |
| 11600000 | 16 |
| 11650000 | 16 |
| 11700000 | 18 |
| 11750000 | 17 |
| 11800000 | 15 |
| 11850000 | 15 |
| 11900000 | 18 |
| 11950000 | 15 |
| 12 000 000 | 430 |
| 12 050 000 | 365 |
| 12100000 | 16 |
| 12 150 000 | 489 |
| 12 200 000 | 311 |
| 12 250 000 | 308 |
| 12 300 000 | 312 |
| 12 350 000 | 338 |
| 12400000 | 18 |
| 12450000 | 20 |
| 12 500 000 | 753 |
| 12 550 000 | 1001 |
| 12 600 000 | 1053 |
| 12650000 | 24 |
| 12700000 | 28 |
| 12 750 000 | 935 |
| 12800000 | 25 |
| 12850000 | 24 |
| 12900000 | 23 |
| 12950000 | 32 |
| 13000000 | 26 |
| 13050000 | 21 |
| 13100000 | 25 |
| 13150000 | 22 |
| 13200000 | 28 |
| 13250000 | 30 |
| 13300000 | 23 |
| 13 350 000 | 694 |
| 13 400 000 | 560 |
| 13 450 000 | 838 |
| 13 500 000 | 587 |
| 13550000 | 19 |
| 13 600 000 | 400 |
| 13650000 | 17 |
| 13700000 | 18 |
| 13 750 000 | 643 |
| 13 800 000 | 701 |
| 13 850 000 | 438 |
| 13 900 000 | 449 |
| 13 950 000 | 432 |
| 14 000 000 | 621 |
| 14 050 000 | 422 |
| 14 100 000 | 474 |
| 14 150 000 | 431 |
| 14 200 000 | 380 |
| 14 250 000 | 535 |
| 14300000 | 19 |
| 14 350 000 | 782 |
| 14 400 000 | 519 |
| 14450000 | 24 |
| 14500000 | 22 |
| 14 550 000 | 493 |
| 14600000 | 19 |
| 14650000 | 20 |
| 14 700 000 | 498 |
| 14 750 000 | 598 |
| 14 800 000 | 501 |
| 14 850 000 | 443 |
| 14 900 000 | 438 |
| 14 950 000 | 617 |
| 15000000 | 24 |
| 15 050 000 | 469 |
| 15 100 000 | 573 |
| 15150000 | 20 |
| 15200000 | 21 |
| 15 250 000 | 435 |
| 15300000 | 24 |
| 15350000 | 21 |
| 15400000 | 19 |
| 15 450 000 | 476 |
| 15 500 000 | 435 |
| 15 550 000 | 495 |
| 15 600 000 | 463 |
| 15650000 | 22 |
| 15 700 000 | 860 |
| 15 750 000 | 471 |
| 15800000 | 20 |
| 15850000 | 23 |
| 15900000 | 22 |
| 15 950 000 | 525 |
| 16 000 000 | 449 |
| 16050000 | 21 |
| 16100000 | 20 |
| 16 150 000 | 907 |
| 16 200 000 | 1336 |
| 16250000 | 28 |
| 16300000 | 24 |
| 16350000 | 26 |
| 16400000 | 24 |
| 16450000 | 22 |
| 16 500 000 | 752 |
| 16 550 000 | 660 |
| 16600000 | 27 |
| 16 650 000 | 1019 |
| 16 700 000 | 648 |
| 16 750 000 | 557 |
| 16800000 | 27 |
| 16850000 | 29 |
| 16900000 | 31 |
| 16 950 000 | 561 |
| 17000000 | 21 |
| 17 050 000 | 463 |
| 17100000 | 22 |
| 17150000 | 26 |
| 17 200 000 | 782 |
| 17250000 | 25 |
| 17 300 000 | 740 |
| 17 350 000 | 834 |
| 17 400 000 | 570 |
| 17 450 000 | 479 |
| 17500000 | 22 |
| 17 550 000 | 543 |
| 17 600 000 | 545 |
| 17650000 | 23 |
| 17700000 | 27 |
| 17750000 | 22 |
| 17800000 | 25 |
| 17850000 | 24 |
| 17 900 000 | 535 |
| 17950000 | 24 |
| 18 000 000 | 480 |
| 18050000 | 25 |
| 18100000 | 23 |
| 18150000 | 25 |
| 18 200 000 | 652 |
| 18 250 000 | 571 |
| 18 300 000 | 511 |
| 18350000 | 23 |
| 18 400 000 | 488 |
| 18 450 000 | 779 |
| 18500000 | 28 |
| 18 550 000 | 674 |
| 18600000 | 30 |
| 18650000 | 24 |
| 18 700 000 | 994 |
| 18750000 | 30 |
| 18 800 000 | 678 |
| 18 850 000 | 591 |
| 18 900 000 | 522 |
| 18 950 000 | 515 |
| 19 000 000 | 547 |
| 19050000 | 24 |
| 19100000 | 27 |
| 19 150 000 | 539 |
| 19 200 000 | 694 |
| 19250000 | 26 |
| 19300000 | 30 |
| 19350000 | 24 |
| 19 400 000 | 680 |
| 19 450 000 | 618 |
| 19 500 000 | 549 |
| 19550000 | 25 |
| 19600000 | 25 |
| 19 650 000 | 552 |
| 19700000 | 27 |
| 19750000 | 30 |
| 19 800 000 | 575 |
| 19 850 000 | 648 |
| 19 900 000 | 889 |
| 19950000 | 27 |
| 20000000 | 26 |
| 20050000 | 35 |
| 20100000 | 36 |
| 20 150 000 | 770 |
| 20200000 | 26 |
| 20 250 000 | 807 |
| 20 300 000 | 833 |
| 20350000 | 28 |
| 20400000 | 28 |
| 20 450 000 | 605 |
| 20500000 | 26 |
| 20 550 000 | 583 |
| 20600000 | 28 |
| 20 650 000 | 635 |
| 20700000 | 28 |
| 20 750 000 | 612 |
| 20 800 000 | 993 |
| 20 850 000 | 749 |
| 20900000 | 30 |
| 20950000 | 32 |
| 21000000 | 28 |
| 21050000 | 31 |
| 21 100 000 | 728 |
| 21150000 | 27 |
| 21200000 | 32 |
| 21250000 | 28 |
| 21300000 | 27 |
| 21 350 000 | 646 |
| 21400000 | 38 |
| 21 450 000 | 742 |
| 21500000 | 29 |
| 21 550 000 | 612 |
| 21600000 | 30 |
| 21650000 | 27 |
| 21 700 000 | 638 |
| 21 750 000 | 645 |
| 21 800 000 | 623 |
| 21 850 000 | 603 |
| 21900000 | 28 |
| 21 950 000 | 699 |
| 22 000 000 | 654 |
| 22 050 000 | 695 |
| 22 100 000 | 1002 |
| 22 150 000 | 730 |
| 22 200 000 | 623 |
| 22 250 000 | 716 |
| 22 300 000 | 602 |
| 22350000 | 29 |
| 22400000 | 29 |
| 22450000 | 28 |
| 22500000 | 30 |
| 22550000 | 29 |
| 22600000 | 32 |
| 22650000 | 33 |
| 22 700 000 | 730 |
| 22750000 | 30 |
| 22800000 | 30 |
| 22 850 000 | 682 |
| 22900000 | 33 |
| 22950000 | 32 |
| 23 000 000 | 618 |
| 23050000 | 33 |
| 23100000 | 32 |
| 23150000 | 31 |
| 23 200 000 | 647 |
| 23250000 | 33 |
| 23 300 000 | 647 |
| 23 350 000 | 675 |
| 23 400 000 | 776 |
| 23 450 000 | 649 |
| 23500000 | 37 |
| 23 550 000 | 667 |
| 23600000 | 31 |
| 23650000 | 31 |
| 23 700 000 | 834 |
| 23750000 | 32 |
| 23800000 | 32 |
| 23850000 | 32 |
| 23900000 | 38 |
| 23 950 000 | 931 |
| 24 000 000 | 1025 |
| 24050000 | 39 |
| 24100000 | 35 |
| 24 150 000 | 842 |
| 24200000 | 34 |
| 24 250 000 | 876 |
| 24 300 000 | 1018 |
| 24 350 000 | 933 |
| 24400000 | 33 |
| 24450000 | 38 |
| 24500000 | 32 |
| 24550000 | 33 |
| 24600000 | 36 |
| 24 650 000 | 755 |
| 24700000 | 34 |
| 24 750 000 | 723 |
| 24 800 000 | 678 |
| 24 850 000 | 687 |
| 24900000 | 34 |
| 24 950 000 | 639 |
| 25 000 000 | 624 |
| 25 050 000 | 689 |
| 25 100 000 | 671 |
| 25 150 000 | 956 |
| 25 200 000 | 695 |
| 25250000 | 36 |
| 25 300 000 | 948 |
| 25 350 000 | 765 |
| 25400000 | 35 |
| 25450000 | 33 |
| 25 500 000 | 914 |
| 25 550 000 | 922 |
| 25600000 | 34 |
| 25 650 000 | 851 |
| 25 700 000 | 854 |
| 25 750 000 | 725 |
| 25800000 | 45 |
| 25 850 000 | 767 |
| 25 900 000 | 723 |
| 25 950 000 | 679 |
| 26000000 | 34 |
| 26 050 000 | 762 |
| 26100000 | 46 |
| 26150000 | 41 |
| 26 200 000 | 884 |
| 26 250 000 | 808 |
| 26300000 | 36 |
| 26 350 000 | 746 |
| 26400000 | 35 |
| 26 450 000 | 743 |
| 26 500 000 | 734 |
| 26550000 | 36 |
| 26 600 000 | 745 |
| 26 650 000 | 928 |
| 26700000 | 45 |
| 26 750 000 | 870 |
| 26 800 000 | 802 |
| 26850000 | 36 |
| 26900000 | 38 |
| 26 950 000 | 842 |
| 27000000 | 37 |
| 27050000 | 34 |
| 27100000 | 36 |
| 27150000 | 40 |
| 27 200 000 | 808 |
| 27250000 | 40 |
| 27 300 000 | 1091 |
| 27350000 | 46 |
| 27400000 | 43 |
| 27 450 000 | 1037 |
| 27 500 000 | 755 |
| 27550000 | 36 |
| 27 600 000 | 855 |
| 27 650 000 | 1076 |
| 27 700 000 | 806 |
| 27 750 000 | 1069 |
| 27 800 000 | 1245 |
| 27850000 | 40 |
| 27900000 | 43 |
| 27 950 000 | 853 |
| 28 000 000 | 777 |
| 28 050 000 | 779 |
| 28 100 000 | 1092 |
| 28 150 000 | 950 |
| 28 200 000 | 959 |
| 28 250 000 | 881 |
| 28 300 000 | 794 |
| 28 350 000 | 844 |
| 28 400 000 | 903 |
| 28 450 000 | 876 |
| 28 500 000 | 901 |
| 28 550 000 | 837 |
| 28 600 000 | 779 |
| 28 650 000 | 1022 |
| 28700000 | 36 |
| 28 750 000 | 982 |
| 28 800 000 | 875 |
| 28 850 000 | 793 |
| 28900000 | 39 |
| 28950000 | 37 |
| 29000000 | 44 |
| 29 050 000 | 814 |
| 29 100 000 | 832 |
| 29150000 | 37 |
| 29 200 000 | 1042 |
| 29250000 | 41 |
| 29300000 | 42 |
| 29350000 | 56 |
| 29400000 | 52 |
| 29450000 | 44 |
| 29500000 | 61 |
| 29550000 | 60 |
| 29 600 000 | 1879 |
| 29 650 000 | 1859 |
| 29 700 000 | 1650 |
| 29750000 | 37 |
| 29 800 000 | 975 |
| 29850000 | 37 |
| 29900000 | 38 |
| 29 950 000 | 879 |
| 30000000 | 38 |
| 30 050 000 | 1088 |
| 30 100 000 | 977 |
| 30150000 | 40 |
| 30200000 | 41 |
| 30 250 000 | 1247 |
| 30300000 | 39 |
| 30350000 | 42 |
| 30400000 | 43 |
| 30450000 | 41 |
| 30500000 | 48 |
| 30 550 000 | 948 |
| 30600000 | 39 |
| 30 650 000 | 895 |
| 30700000 | 42 |
| 30 750 000 | 912 |
| 30800000 | 40 |
| 30 850 000 | 917 |
| 30900000 | 48 |
| 30950000 | 55 |
| 31 000 000 | 978 |
| 31 050 000 | 965 |
| 31 100 000 | 992 |
| 31 150 000 | 885 |
| 31200000 | 50 |
| 31250000 | 45 |
| 31300000 | 42 |
| 31 350 000 | 942 |
| 31 400 000 | 866 |
| 31 450 000 | 903 |
| 31500000 | 45 |
| 31550000 | 39 |
| 31 600 000 | 936 |
| 31650000 | 43 |
| 31700000 | 42 |
| 31750000 | 40 |
| 31800000 | 42 |
| 31 850 000 | 1008 |
| 31 900 000 | 1259 |
| 31 950 000 | 933 |
| 32000000 | 41 |
| 32 050 000 | 960 |
| 32 100 000 | 1026 |
| 32 150 000 | 1095 |
| 32 200 000 | 1249 |
| 32 250 000 | 1251 |
| 32 300 000 | 1150 |
| 32 350 000 | 1228 |
| 32 400 000 | 1174 |
| 32 450 000 | 1287 |
| 32500000 | 50 |
| 32 550 000 | 1142 |
| 32 600 000 | 1034 |
| 32650000 | 52 |
| 32 700 000 | 1005 |
| 32750000 | 46 |
| 32800000 | 43 |
| 32 850 000 | 1360 |
| 32900000 | 63 |
| 32950000 | 63 |
| 33 000 000 | 1146 |
| 33 050 000 | 943 |
| 33 100 000 | 1014 |
| 33 150 000 | 1115 |
| 33200000 | 43 |
| 33 250 000 | 1086 |
| 33 300 000 | 936 |
| 33350000 | 43 |
| 33400000 | 48 |
| 33 450 000 | 1097 |
| 33500000 | 48 |
| 33550000 | 45 |
| 33 600 000 | 4498 |
| 33 650 000 | 1423 |
| 33 700 000 | 970 |
| 33 750 000 | 940 |
| 33800000 | 44 |
| 33850000 | 46 |
| 33900000 | 43 |
| 33950000 | 47 |
| 34 000 000 | 1036 |
| 34050000 | 47 |
| 34100000 | 43 |
| 34 150 000 | 953 |
| 34200000 | 49 |
| 34 250 000 | 1077 |
| 34 300 000 | 1441 |
| 34 350 000 | 1155 |
| 34 400 000 | 1594 |
| 34 450 000 | 1573 |
| 34 500 000 | 1233 |
| 34550000 | 45 |
| 34 600 000 | 1214 |
| 34 650 000 | 1051 |
| 34700000 | 43 |
| 34 750 000 | 1077 |
| 34800000 | 49 |
| 34 850 000 | 1021 |
| 34900000 | 48 |
| 34 950 000 | 949 |
| 35000000 | 47 |
| 35 050 000 | 1058 |
| 35 100 000 | 1106 |
| 35150000 | 88 |
| 35200000 | 51 |
| 35250000 | 58 |
| 35 300 000 | 1399 |
| 35 350 000 | 1367 |
| 35400000 | 49 |
| 35450000 | 46 |
| 35 500 000 | 2272 |
| 35550000 | 55 |
| 35 600 000 | 1857 |
| 35 650 000 | 1523 |
| 35700000 | 56 |
| 35750000 | 54 |
| 35800000 | 57 |
| 35 850 000 | 1498 |
| 35900000 | 60 |
| 35950000 | 61 |
| 36 000 000 | 2116 |
| 36 050 000 | 2196 |
| 36100000 | 96 |
| 36150000 | 84 |
| 36 200 000 | 3124 |
| 36250000 | 72 |
| 36 300 000 | 2431 |
| 36350000 | 74 |
| 36400000 | 75 |
| 36450000 | 57 |
| 36500000 | 59 |
| 36550000 | 53 |
| 36 600 000 | 1512 |
| 36650000 | 53 |
| 36 700 000 | 1373 |
| 36750000 | 59 |
| 36 800 000 | 1450 |
| 36850000 | 50 |
| 36 900 000 | 1304 |
| 36950000 | 59 |
| 37000000 | 55 |
| 37050000 | 68 |
| 37 100 000 | 1971 |
| 37150000 | 60 |
| 37 200 000 | 1745 |
| 37250000 | 62 |
| 37 300 000 | 1428 |
| 37350000 | 55 |
| 37 400 000 | 1397 |
| 37450000 | 54 |
| 37 500 000 | 1406 |
| 37 550 000 | 1159 |
| 37600000 | 51 |
| 37 650 000 | 1220 |
| 37700000 | 56 |
| 37750000 | 53 |
| 37 800 000 | 1196 |
| 37850000 | 48 |
| 37 900 000 | 1116 |
| 37950000 | 51 |
| 38 000 000 | 1437 |
| 38 050 000 | 1292 |
| 38100000 | 61 |
| 38 150 000 | 1276 |
| 38200000 | 51 |
| 38250000 | 48 |
| 38300000 | 47 |
| 38 350 000 | 1111 |
| 38400000 | 62 |
| 38450000 | 83 |
| 38 500 000 | 1843 |
| 38 550 000 | 1577 |
| 38600000 | 64 |
| 38 650 000 | 1574 |
| 38 700 000 | 1676 |
| 38750000 | 56 |
| 38 800 000 | 1538 |
| 38 850 000 | 1397 |
| 38900000 | 60 |
| 38950000 | 62 |
| 39000000 | 51 |
| 39 050 000 | 1222 |
| 39100000 | 56 |
| 39150000 | 56 |
| 39200000 | 52 |
| 39250000 | 53 |
| 39300000 | 56 |
| 39 350 000 | 1192 |
| 39 400 000 | 1197 |
| 39450000 | 52 |
| 39 500 000 | 1323 |
| 39550000 | 51 |
| 39 600 000 | 1252 |
| 39650000 | 52 |
| 39 700 000 | 1222 |
| 39 750 000 | 1166 |
| 39800000 | 52 |
| 39850000 | 56 |
| 39900000 | 66 |
| 39950000 | 56 |
| 40000000 | 55 |
| 40 050 000 | 1221 |
| 40 100 000 | 1273 |
| 40 150 000 | 1269 |
| 40 200 000 | 1272 |
| 40 250 000 | 1213 |
| 40 300 000 | 1164 |
| 40 350 000 | 1083 |
| 40400000 | 55 |
| 40450000 | 59 |
| 40500000 | 63 |
| 40 550 000 | 1179 |
| 40600000 | 58 |
| 40650000 | 58 |
| 40 700 000 | 1273 |
| 40750000 | 53 |
| 40800000 | 53 |
| 40850000 | 56 |
| 40900000 | 53 |
| 40 950 000 | 1291 |
| 41 000 000 | 1794 |
| 41050000 | 58 |
| 41100000 | 57 |
| 41150000 | 62 |
| 41 200 000 | 1198 |
| 41 250 000 | 1157 |
| 41 300 000 | 1117 |
| 41350000 | 54 |
| 41400000 | 55 |
| 41450000 | 54 |
| 41 500 000 | 1157 |
| 41 550 000 | 1220 |
| 41600000 | 54 |
| 41 650 000 | 1268 |
| 41 700 000 | 1546 |
| 41 750 000 | 1244 |
| 41800000 | 54 |
| 41850000 | 58 |
| 41900000 | 60 |
| 41 950 000 | 1218 |
| 42 000 000 | 1295 |
| 42050000 | 53 |
| 42100000 | 71 |
| 42150000 | 68 |
| 42 200 000 | 1413 |
| 42250000 | 61 |
| 42 300 000 | 1315 |
| 42 350 000 | 1222 |
| 42 400 000 | 1249 |
| 42 450 000 | 1267 |
| 42500000 | 54 |
| 42 550 000 | 1323 |
| 42 600 000 | 1426 |
| 42650000 | 58 |
| 42 700 000 | 1623 |
| 42 750 000 | 1507 |
| 42800000 | 69 |
| 42 850 000 | 1557 |
| 42900000 | 64 |
| 42 950 000 | 1547 |
| 43 000 000 | 1536 |
| 43 050 000 | 1513 |
| 43 100 000 | 1516 |
| 43150000 | 66 |
| 43 200 000 | 1726 |
| 43 250 000 | 1859 |
| 43 300 000 | 2283 |
| 43350000 | 65 |
| 43400000 | 65 |
| 43450000 | 60 |
| 43500000 | 62 |
| 43550000 | 60 |
| 43 600 000 | 1924 |
| 43 650 000 | 1526 |
| 43700000 | 62 |
| 43 750 000 | 1506 |
| 43 800 000 | 1521 |
| 43850000 | 62 |
| 43900000 | 64 |
| 43950000 | 66 |
| 44 000 000 | 1684 |
| 44050000 | 72 |
| 44100000 | 72 |
| 44 150 000 | 1583 |
| 44 200 000 | 1557 |
| 44 250 000 | 1424 |
| 44 300 000 | 1225 |
| 44350000 | 56 |
| 44400000 | 56 |
| 44 450 000 | 1229 |
| 44 500 000 | 1256 |
| 44 550 000 | 1337 |
| 44 600 000 | 1238 |
| 44650000 | 61 |
| 44700000 | 59 |
| 44750000 | 65 |
| 44800000 | 65 |
| 44850000 | 61 |
| 44 900 000 | 1338 |
| 44950000 | 63 |
| 45 000 000 | 1282 |
| 45050000 | 59 |
| 45100000 | 56 |
| 45150000 | 65 |
| 45200000 | 57 |
| 45 250 000 | 1304 |
| 45 300 000 | 1262 |
| 45350000 | 61 |
| 45400000 | 64 |
| 45450000 | 67 |
| 45500000 | 64 |
| 45550000 | 64 |
| 45 600 000 | 1284 |
| 45650000 | 56 |
| 45 700 000 | 1258 |
| 45 750 000 | 1290 |
| 45800000 | 63 |
| 45850000 | 58 |
| 45900000 | 57 |
| 45 950 000 | 1295 |
| 46000000 | 59 |
| 46050000 | 59 |
| 46 100 000 | 1568 |
| 46150000 | 69 |
| 46 200 000 | 1916 |
| 46250000 | 64 |
| 46300000 | 76 |
| 46 350 000 | 2095 |
| 46 400 000 | 1941 |
| 46 450 000 | 1946 |
| 46 500 000 | 1479 |
| 46 550 000 | 1519 |
| 46600000 | 69 |
| 46650000 | 75 |
| 46 700 000 | 1501 |
| 46 750 000 | 1401 |
| 46800000 | 59 |
| 46 850 000 | 1299 |
| 46900000 | 58 |
| 46950000 | 59 |
| 47000000 | 58 |
| 47 050 000 | 1285 |
| 47 100 000 | 1213 |
| 47150000 | 65 |
| 47 200 000 | 1756 |
| 47 250 000 | 1876 |
| 47 300 000 | 1642 |
| 47 350 000 | 1508 |
| 47400000 | 92 |
| 47450000 | 96 |
| 47 500 000 | 2113 |
| 47 550 000 | 113 |
| 47600000 | 91 |
| 47 650 000 | 1881 |
| 47700000 | 86 |
| 47750000 | 92 |
| 47 800 000 | 1628 |
| 47 850 000 | 1485 |
| 47900000 | 94 |
| 47 950 000 | 1537 |
| 48000000 | 90 |
| 48 050 000 | 1556 |
| 48100000 | 96 |
| 48 150 000 | 1547 |
| 48200000 | 99 |
| 48 250 000 | 1552 |
| 48 300 000 | 106 |
| 48350000 | 93 |
| 48 400 000 | 1482 |
| 48 450 000 | 1642 |
| 48 500 000 | 1506 |
| 48 550 000 | 1499 |
| 48600000 | 98 |
| 48650000 | 87 |
| 48 700 000 | 1578 |
| 48 750 000 | 1458 |
| 48 800 000 | 101 |
| 48 850 000 | 1631 |
| 48 900 000 | 144 |
| 48 950 000 | 1883 |
| 49 000 000 | 122 |
| 49 050 000 | 1799 |
| 49 100 000 | 1883 |
| 49 150 000 | 132 |
| 49200000 | 91 |
| 49 250 000 | 1798 |
| 49 300 000 | 127 |
| 49 350 000 | 166 |
| 49 400 000 | 145 |
| 49 450 000 | 1722 |
| 49 500 000 | 1668 |
| 49 550 000 | 117 |
| 49600000 | 93 |
| 49650000 | 96 |
| 49 700 000 | 100 |
| 49 750 000 | 124 |
| 49 800 000 | 1745 |
| 49 850 000 | 123 |
| 49 900 000 | 2737 |
| 49 950 000 | 125 |
| 50 000 000 | 138 |