МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение   
высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА 41

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Асп. |  |  |  | К.Н. Смирнов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКАК КУРСОВОЙ РАБОТЕ |
| РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА И ФУНКЦИЙ ДЛЯ РАБОТЫ СО СТРУКТУРОЙ ДАННЫХ ТИПА «ОЧЕРЕДЬ» В MATLAB |
| по дисциплине: ТЕХНОЛОГИИ ПРОГРАМИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛА

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТКА ГР. | 4317КС |  |  |  | А. С. Турчина |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2015

Оглавление

[Постановка задачи 3](#_Toc462153591)

[Графический интерфейс пользователя 3](#_Toc462153592)

[Описание структур данных используемых в программе 4](#_Toc462153593)

[Листинг программы 8](#_Toc462153594)

[Инструкция по работе с программой 14](#_Toc462153595)

[Заключение 17](#_Toc462153596)

[Список литературы 18](#_Toc462153597)

# Постановка задачи

Разработать графический интерфейс с реализацией работы функций для структуры данных типа «Очередь» в MATLAB согласно своему варианту.

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Объект | Способ организации данных | Реализуемая функция |
| 9 | Данные автомобиля | Очередь | Вставка элемента в хвост |

Для создания базы данных автомобиля необходимо создать очередь. Предполагается хранить следующие параметры автомобиля:

* Идентификационный номер (ID)
* Фирма изготовитель (Make)
* Модель (Model)
* Год производства (Year)
* Тип трансмиссии (Transmission)
* Тип двигателя (Engine)
* Тип привода (Drive type)
* Цвет кузова (Color)

Реализовать будем следующие функции:

* Добавление нового автомобиля в конец очереди
* Извлечение автомобиля из начала очереди
* Проверка на пустоту очереди
* Получение количества автомобилей в очереди

# Графический интерфейс пользователя

Графический интерфейс пользователя реализован в файле menu.m, листинг которого приведен в главе «Листинг программы». графический интерфейс позволяет добавлять элементы в очередь и извлекать их из очереди.

GUI создан программно, т.е. без использования утилиты GUIDE [1]

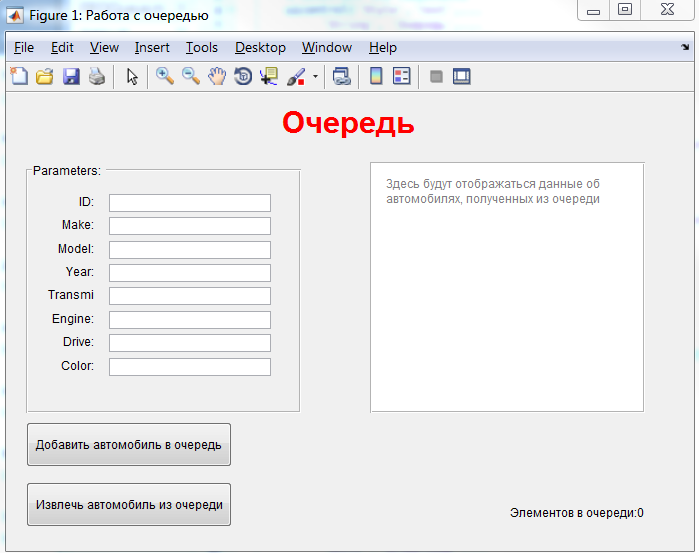


Рис. 1 Графический интерфейс

# Описание структур данных используемых в программе

Использовался способ связи – очередь.

**«**Очередь илиFIFO (акроним First In, First Out – «первым пришёл – первым ушёл») – способ организации и манипулирования данными относительно времени. Это выражение описывает принцип технической обработки очереди или обслуживания конфликтных требований путём упорядочения процесса по принципу: «первым пришёл – первым обслужен» (ПППО). Тот, кто приходит первым, тот и обслуживается первым, пришедший следующим ждёт, пока обслуживание первого не будет закончено, и так далее.»[2]

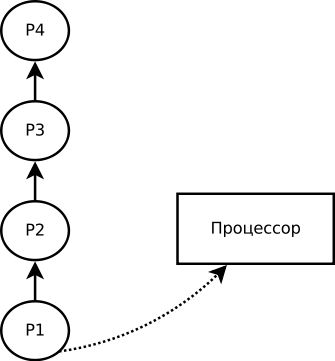


Рис. Граф очереди

Каждый элемент очереди можно представить вершиной графа. Вершины соединены направленными дугами. Начальный узел очереди называют «головой», конечный – «хвостом». Из головы очереди выходит одна дуга и ни одной не входит. Хвост очереди напротив же не имеет выходящих дуг и имеет одну входящую. Все остальные вершины имеют по одной входящей и одной выходящей дуге.

Основное свойство очереди заключено в принципе FIFO – First In Firs Out. Очередь часто используется в качестве буфера для хранения потока поступающих заявок. Хороший пример очереди – обычная очередь к кассе. Каждый покупатель является элементом и обслуживается в соответствии с принципом FIFO, т.е. тот, кто раньше пришел, раньше будет обслужен.

Важный параметр – это длина очереди. Длина очереди равна количеству элементов в очереди. Если длина очереди ограничена, то при переполнении новые элементы не ставятся в очередь.

В данной работе мы будем организовывать хранение данных следующим способом. Класс FIFOQueue представляющий собой очередь. Данный класс имеет следующие атрибуты:

**element** – массив ячеек, который хранит ссылки на объекты, являющиеся элементами очереди

**head** – указатель ячейку в массиве element, в которой хранится головной элемент очереди

**tail** – указатель ячейку в массиве element, следующую за ячейкой с хвостовым элементом. Т.е. если хвостовой элемент хранится в ячейке 3, то tail=4

**count** – количество элементов очереди. Вычисляется как разность между head и tail

Массив ячеек element имеет конечную величину. Можно закольцевать массив, так что при достижение конца очереди, указатель переводится в начало массива циклически, однако при этом длина очереди будет ограничена длиной массива [3]. Мы поступим по-другому. При заполнении массива мы будем выделять дополнительную память. Один из самых простых способов, это при переполнении element добавить к нему фиксированное число пустых ячеек.

Однако при последующем извлечении элементов из головы очереди возникает проблема, первые ячейки массива становятся ненужными, но все также занимают место в памяти. Поэтому при достижении указателем head некоторого фиксированного значения REFLENGTH, мы копируем хвостовую часть массива ячеек от head до конца в начало массива. Таким образом head становится равным 1, а tail уменьшается на REFLENGTH. Ввиду того, что в массиве ячеек element хранятся не сами объекты класса Car, а лишь ссылки на них, то копирование не занимает много времени.

Для работы с очередью реализованы следующие методы класса FIFOQueue

**enqueue(elment)** – метод добавляет элемент element в хвост списка. Соответственно при каждом добавлении, мы увеличиваем указатель tail на 1 и счетчик count тоже на 1. Кроме того, при достижении указателем tail границы массива ячеек element. Мы увеличиваем данный массив на фиксированную величину REFLENGTH.

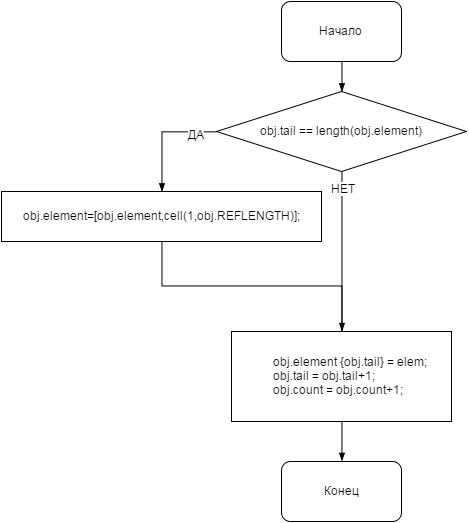


Рис. 3 Блок-схема метода enqueue

**element = dequeue()** – метод извлекает ссылку на элемент из головы списка и возвращает его. Соответственно, при каждом извлечении элемента, указатель head увеличивается на 1, а счетчик count уменьшается на 1. При достижении указателем head величины REFLENGTH происходит уменьшение контейнера element по алгоритму.

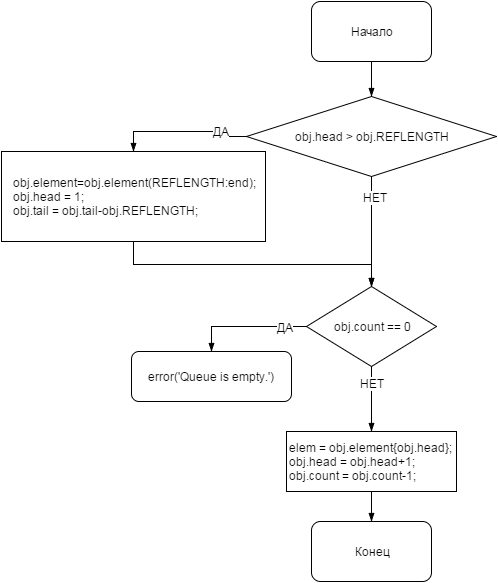


Рис. Блок-схема метода dequeue

**result = isempty()** – метод возвращает true, если очередь пуста. Т.е. если count равен нулю, иначе возвращается false

**cnt = get\_count()** – метод возвращает количество элементов в очереди

Наша задача – хранить в очереди параметры автомобиля. В качестве контейнера для хранения параметров был реализован класс Car. Его атрибуты:

* Идентификационный номер (ID)
* Фирма изготовитель (Make)
* Модель (Model)
* Год производства (Year)
* Тип трансмиссии (Transmission)
* Тип двигателя (Engine)
* Тип привода (Drive type)
* Цвет кузова (Color)

При вызове конструктора класса Car, необходимо указать id автомобиля.

Для добавления остальных атрибутов реализован метод **addfield(varargin)**, где varargin это массив ячеек с атрибутами и их значениями. Это позволяет задать сразу несколько атрибутов за один вызов метода.

Пример использования:

% Создаем автомобиль

car=Car(1);

% Добавляем параметры автомобиля

car.addfield('make','bmw',...

'model','m3',...

'year',2016,...

'transmission','8AT',...

'engine','5.0',...

'drive\_type','RWD',...

'color','red');

Также реализован метод **delfield(varargin)**, где varargin массив атрибутов которые нужно очистить.

Пример:

car.delfield('make', 'color');

Кроме самих структур данных была реализована функция **addtail(queue, varargin)**. Данная функция добавляет в конец очереди сразу несколько элементов. Ее листинг приведен в разделе «Листинг программы», а блок-схема приведена ниже.

Таким образом для реализации программы в Matlab было написано два класса: Car и FIFOQueue. Экземпляр класса Car хранит свойства автомобиля, описанные выше. Экземпляр класса FIFOQueue является очередью, которая реализует хранение ссылок на любые типа объектов. В нашем случаев в качестве элементов очереди будут выступать ссылки на экземпляры класса Car.

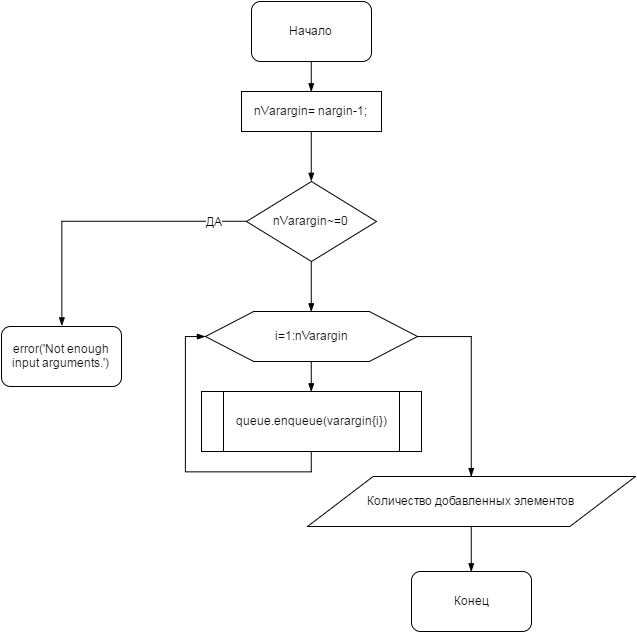


Рис. Блок-схема функции addtail

# Листинг программы

menu.m

|  |  |
| --- | --- |
|  | function menu  % Create and then hide the GUI as it is being constructed.  f = figure('Visible','off','Position',[360,500,600,400]);    % Construct the components  uicontrol('Style','text',...  'String', 'Очередь',...  'ForegroundColor',[1 0 0],...  'FontSize',20,...  'FontWeight','bold',...  'HorizontalAlignment','center',...  'Units','normalized',...  'Position',[.1 .9 .8 .08]);    hpanel = uipanel('Position',[.03 .3 .4 .55],'Title','Parameters:');  hpanel2 = uipanel('Position',[.53 .3 .4 .55],...  'BackgroundColor','white');  hresult\_text = uicontrol('Parent', hpanel2,...  'Style','text',...  'String', 'Здесь будут отображаться данные об автомобилях, полученных из очереди',...  'ForegroundColor',[0.5 0.5 0.5],...  'HorizontalAlignment','left',...  'Units','normalized',...  'Position',[.05 .05 .9 .9],...  'BackgroundColor','white');    hcount = uicontrol('Style','text',...  'String',strcat('Элементов в очереди: 0',...  'Units','normalized',...  'HorizontalAlignment','right',...  'Position',[.53 .05 .4 .05]);    hid\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'ID:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .85 .2 .08]);  hid = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .85 .6 .08]);    hmake\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Make:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .75 .2 .08]);  hmake = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .75 .6 .08]);    hmodel\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Model:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .65 .2 .08]);  hmodel = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .65 .6 .08]);    hyear\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Year:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .55 .2 .08]);  hyear = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .55 .6 .08]);    htransmission\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Transmission:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .45 .2 .08]);  htransmission = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .45 .6 .08]);    hengine\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Engine:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .35 .2 .08]);  hengine = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .35 .6 .08]);    hdrive\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Drive:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .25 .2 .08]);  hdrive = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .25 .6 .08]);    hcolor\_text = uicontrol('Parent', hpanel,...  'Style','text',...  'String', 'Color:',...  'HorizontalAlignment','right',...  'Units','normalized',...  'Position',[.05 .15 .2 .08]);  hcolor = uicontrol('Parent', hpanel,...  'Style','edit',...  'Units','normalized',...  'Position',[.3 .15 .6 .08]);  hadd\_button = uicontrol('Style','pushbutton',...  'String','Добавить автомобиль в очередь',...  'Units','normalized',...  'Position',[.03 .18 .3 .1],...  'Callback',@add\_button\_callback);  hget\_button = uicontrol('Style','pushbutton',...  'String','Извлечь автомобиль из очереди',...  'Units','normalized',...  'Position',[.03 .05 .3 .1],...  'Callback',@get\_button\_callback);    q = FIFOQueue;    % Assign the GUI a name to appear in the window title.  f.Name = 'Работа с очередью';    % Move the GUI to the center of the screen.  movegui(f,'center')    % Make the GUI visible.  f.Visible = 'on';    % Callbacks  function add\_button\_callback(source,eventdata)  hresult\_text.ForegroundColor = [0 0 0];  if isempty(hid.String)  hresult\_text.ForegroundColor = [1 0 0];  hresult\_text.String = 'Поле ID обязательное для ввода';  return  end  car = Car(str2double(hid.String));  car.addfield('make',hmake.String,...  'model',hmodel.String,...  'year',str2double(hyear.String),...  'transmission',htransmission.String,...  'engine',hengine.String,...  'drive\_type',hdrive.String,...  'color',hcolor.String);  addtail(q,car)  hid.String = '';  hmake.String = '';  hmodel.String = '';  hyear.String = '';  htransmission.String = '';  hengine.String = '';  hdrive.String = '';  hcolor.String = '';    hresult\_text.String = 'Данные добавлены в очередь';  hcount.String = strcat('Элементов в очереди: ',num2str(q.get\_count));  end    function get\_button\_callback(source,eventdata)  count = q.get\_count;  hresult\_text.ForegroundColor = [0 0 0];  if count==0  hresult\_text.ForegroundColor = [1 0 0];  hresult\_text.String = 'В очереди нет элементов';  return  end  car = q.dequeue;  result\_text = sprintf('ID: %d\nMake: %s\nModel: %s\nYear: %d\nTransm.: %s\nEngine: %s\nDrive: %s\nColor: %s\n',...  car.id,car.make,car.model,car.year,car.transmission,car.engine,car.drive\_type,car.color);  hresult\_text.String = result\_text;  hcount.String = strcat('Элементов в очереди: ',num2str(q.get\_count));  end      end |

Car.m

|  |  |
| --- | --- |
|  | classdef Car < handle  % Реализация класса Car для Хранения параметров автомобилей  properties  id  make  model  year  transmission  engine  drive\_type  color  end    methods  function obj = Car(id)  % Конструктор  obj.id = id;  end    function addfield(obj,varargin)  % Добавление полей  nVarargin = nargin-1;  if ~nVarargin || rem(nVarargin,2)~=0  error('Not enough input arguments.')  end  i=1;  while i<=nVarargin  obj.(varargin{i}) = varargin{i+1};  i = i+2;  end    end    function delfield(obj,varargin)  % Очистка полей  nVarargin = nargin-1;  if ~nVarargin  error('Not enough input arguments.')  end  i=1;  while i<=nVarargin  obj.(varargin{i}) = [];  i = i+1;  end  end  end  end |

FIFOQueue.m

|  |  |
| --- | --- |
|  | classdef FIFOQueue < handle    properties(Access=private, Constant=true)  REFLENGTH = 1000;  end    properties  head  tail  count  element  end    methods  function obj = FIFOQueue()  % Конструктор очереди  obj.element = cell(1,obj.REFLENGTH);  obj.head = 1;  obj.tail = 1;  obj.count = 0;  end    function enqueue(obj, elem)  % Вставка элемента в хвост очереди    % Увеличиваем контейнер при заполнении  if obj.tail == length(obj.element)  obj.element=[obj.element,cell(1,obj.REFLENGTH)];  end  obj.element{obj.tail} = elem;  obj.tail = obj.tail+1;  obj.count = obj.count+1;  end    function elem = dequeue(obj)  % Получение элемента из головы очереди    % Уменьшаем контейнер при head > REFLENGTH  if obj.head > obj.REFLENGTH  obj.element=obj.element(11:end);  obj.head = 1;  obj.tail = obj.tail-obj.REFLENGTH;  end  if obj.count == 0  error('Queue is empty.')  end  elem = obj.element{obj.head};  obj.head = obj.head+1;  obj.count = obj.count-1;  end    function result=isempty(obj)  % Проверка очереди на пустоту  result = obj.count==0;  end    function cnt = get\_count(obj)  % Возвращает количество элементов в очереди  cnt = obj.count;  end  end    end |

addtail.m

|  |  |
| --- | --- |
|  | function addtail(queue, varargin)  % Добавление нескольких элементов в конец очереди  nVarargin = nargin-1;  if ~nVarargin  error('Not enough input arguments.')  end  for i=1:nVarargin  queue.enqueue(varargin{i})  end  %fprintf('Количество добавленных элементов = %d \n',nVarargin) |

# Инструкция по работе с программой

При запуске графического интерфейса программы пользователь видит следующее окно:

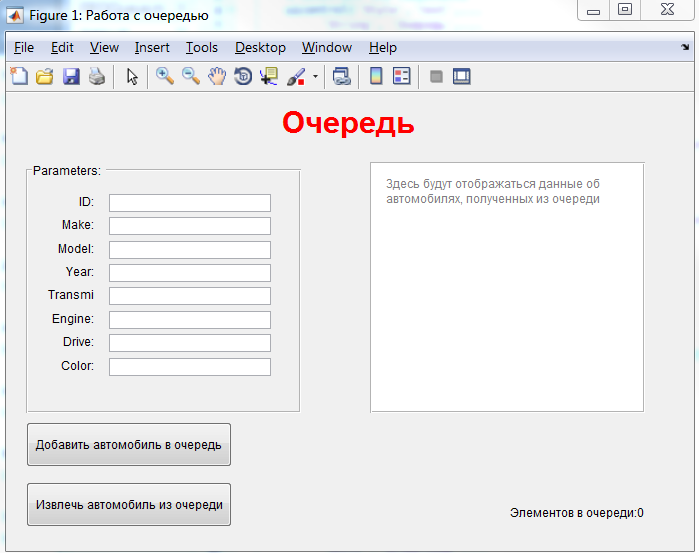


Рис. Стартовое окно

Для добавления автомобиля в очередь необходимо ввести параметры автомобиля в соответствующие поля и нажать кнопку «Добавить автомобиль в очередь».

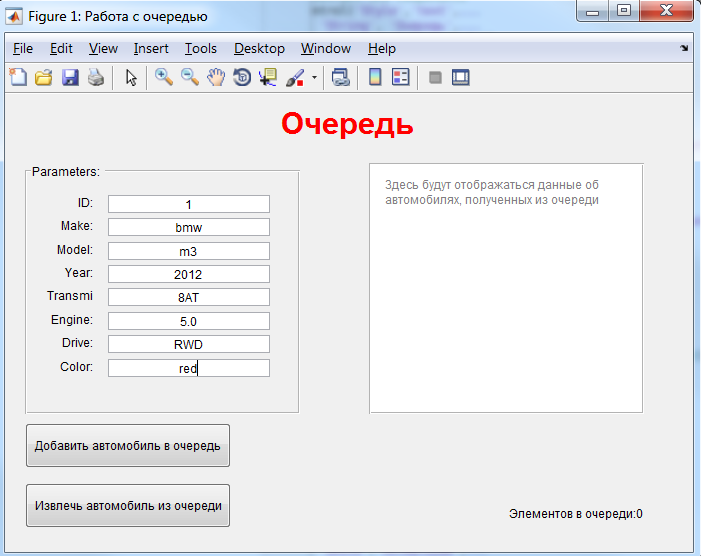


Рис. 7 Добавление автомобиля в очередь

Обязательно для заполнения только поле ID. Остальные поля могут быть пустыми. Если не заполнить поле ID, то будет выведено предупреждение.

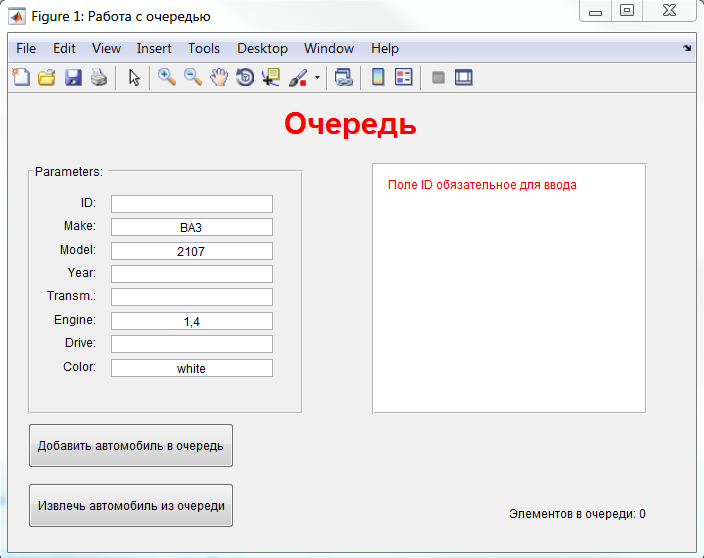


Рис. Предупреждение о пустом поле ID

После добавления автомобиля в очередь поля с параметрами будут очищены. Количество добавленных элементов можно отслеживать в счетчике в правом нижнем углу окна. В дальнейшем можно добавить следующий автомобиль либо извлечь элемент из очереди.

Извлечение элемента осуществляется нажатием кнопки «Извлечь автомобиль из очереди»

Параметры извлеченного автомобиля выводятся в окне:

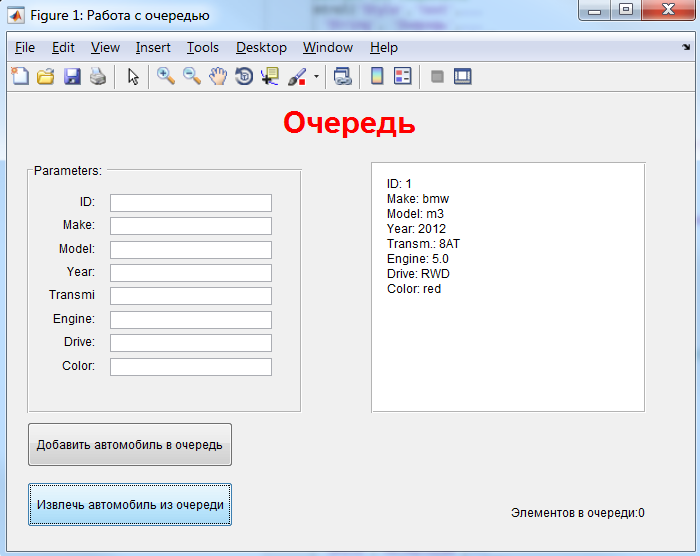


Рис. Извлечение автомобиля из очереди

Если пользователь попытается извлечь элемент из пустой очереди, он увидит предупреждение:

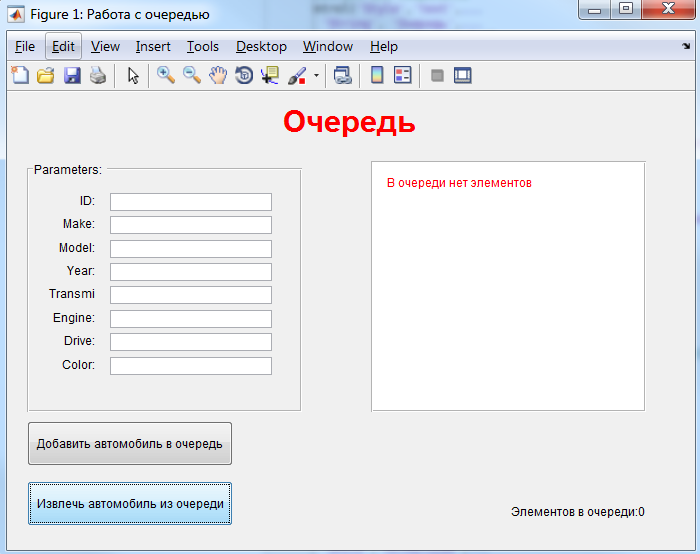


Рис. Предупреждение о пустой очереди

# Заключение

В данной курсовой работе была продемонстрирована работа графического интерфейса с реализацией функций для работы со структурой данных типа «Очередь».

# Список литературы

1. <http://www.mathworks.com/help/matlab/creating_guis/about-the-simple-programmatic-gui-example.html>
2. <https://ru.wikipedia.org/wiki/FIFO>
3. <http://www.intuit.ru/studies/courses/2193/67/lecture/1980?page=3>