**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

Кафедра 41

Преподаватель

Асп. Смирнов К. Н.

Отчет

по лабораторной работе

по дисциплине ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

на тему: «Структуры данных»

Работу выполнила

студентка гр. 4317КС Турчина Анна Станиславовна

СПб

2015 г.

*Цель работы*: описать структуру данных, соответствующую заданному объекту. Реализовать функцию по работе с данными, организованными в указанном виде.

*Вариант задания*:

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

1. Описание структуры данных

Сначала создаем класс Car, который будет хранить параметры автомобиля:

classdef Car < handle

% Реализация класса Саг 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

Далее создаем класс очереди (FIFO). Очередь будет хранить ссылки на объекты класса Car.

classdef FIFOQueue < handle

properties(Access=private, Constant=true)

REFLENGTH = 1000;

end

properties(Access=private)

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-100;

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

end

end

Создаем тестовую очередь

% Создаем очередь

q=FIFOQueue;

% 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');

% Добавляем автомобиль в очередь

q.enqueue(car);

% Аналогично создаем другие автомобили и добавляем их в очередь

car=Car(2);

car.addfield('make','ВАЗ',...

'model','2107',...

'year',1995,...

'transmission','4MT',...

'engine','1.4',...

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

'color','white');

q.enqueue(car);

car=Car(3);

car.addfield('make','Nissan',...

'model','Juke',...

'year',2011,...

'transmission','CVT',...

'engine','1.6T',...

'drive\_type','AWD',...

'color','black');

q.enqueue(car);

car=Car(4);

car.addfield('make','Tesla',...

'model','ModelS',...

'year',2015,...

'transmission',[],...

'engine','Electro',...

'drive\_type','AWD',...

'color','red');

q.enqueue(car);

car=Car(5);

car.addfield('make','Hyundai',...

'model','Creta',...

'year',2016,...

'transmission','6AT',...

'engine','1.6 CRDi VGT',...

'drive\_type','AWD',...

'color','black');

q.enqueue(car);

car=Car(6);

car.addfield('make','Ford',...

'model','Model T',...

'year',1908,...

'transmission','2PG',...

'engine','2.9',...

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

'color','black');

q.enqueue(car);

car=Car(7);

car.addfield('make','ВАЗ',...

'model','Priora',...

'year',2010,...

'transmission','5MT',...

'engine','2.9',...

'drive\_type','FWD',...

'color','white');

q.enqueue(car);

2. Функция добавления элемента в конец очереди

Добавление элемента в конец очереди реализовано в классе FIFOQueue в качестве метода enqueue(). Ниже приведен сам метод и его блок-схема. Также реализована функция addtail, которая добавляет в конец очереди сразу несколько элементов. Ее листинг и блок-схема также приведены ниже.

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

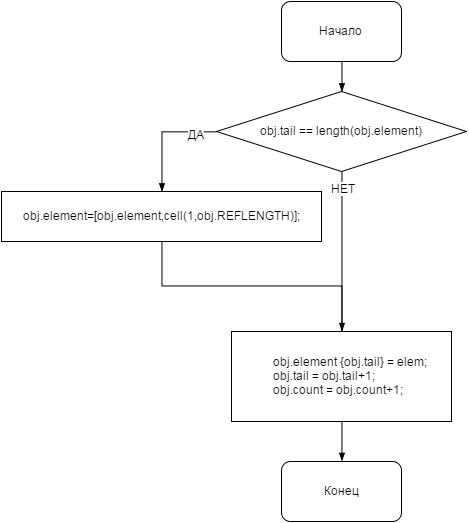


Рис. 1 Блок схема метода enqueue

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)

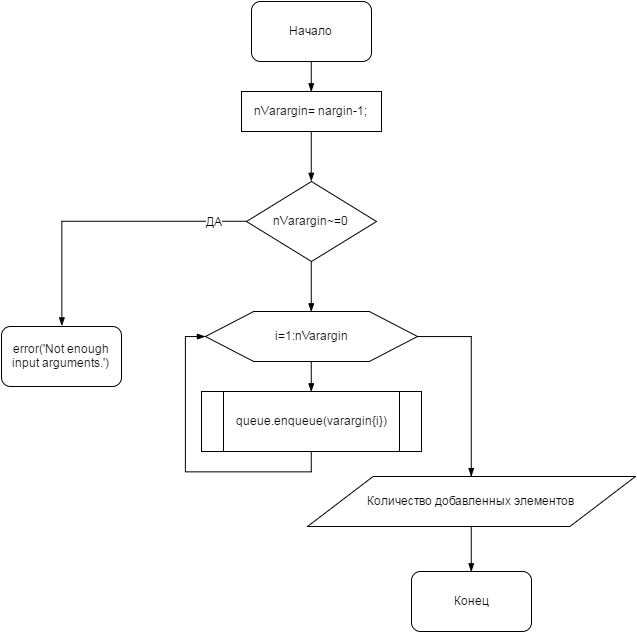


Рис. 2 Блок-схема функции addtail