МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной работе №3
«Динамические списки.»
По дисциплине «Л и ОА в ИЗ»

Выполнили: ст. гр. 22ВВ4
Жуков Илья
Чумаев Сабит

Приняли: Юрова О.В.
Акифьев И.В.

2023

Цель работы:

Написать код программы, выполнив следующие задания:

1.Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2.* На основе приведенного кода реализуйте структуру данных *Очередь*.
3.* На основе приведенного кода реализуйте структуру данных *Стек*.

Ход работы:

Описание кода программы 1:

В данной программе реализуется приоритетная очередь, путем добавления элемента в список в соответствии с приоритетом объекта

```
using System;

using System.Collections.Generic;


public class PriorityQueue<T>

{

    //объявляет приватное поле queue типа List<T> для хранения
элементов очереди

    private List<T> queue;

    //объявляет приватное поле priorities типа List<int> для хранения
приоритетов элементов очереди.

    private List<int> priorities;


    //Создает новые экземпляры List<T> и List<int> для queue и
priorities соответственно.

    public PriorityQueue()

    {

        queue = new List<T>();

        priorities = new List<int>();

    }
```

```
    //метод для добавления элемента item с приоритетом priority в очередь.
    //Он находит правильную позицию в очереди, основываясь на приоритете, и вставляет элемент и приоритет в соответствующие списки.
    public void Enqueue(T item, int priority)
    {
        int index = queue.Count;
        while (index > 0 && priority <= priorities[index - 1])
        {
            index--;
        }

        queue.Insert(index, item);
        priorities.Insert(index, priority);
    }

    //метод для удаления и возврата первого элемента очереди
    public T Dequeue()
    {
        //Он выбрасывает исключение InvalidOperationException, если очередь пуста.
        if (queue.Count == 0)
            throw new InvalidOperationException("Queue is empty");

        T item = queue[0];
        queue.RemoveAt(0);
        priorities.RemoveAt(0);
        return item;
    }
```

```
    //свойство, возвращающее количество элементов в очереди.
    public int Count
    {
        get { return queue.Count; }
    }

    //свойство, возвращающее значение true, если очередь пуста, и
false в противном случае.
    public bool IsEmpty
    {
        get { return queue.Count == 0; }
    }

    public void Clear()
    {
        queue.Clear();
        priorities.Clear();
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        PriorityQueue<string> queue = new PriorityQueue<string>();

        Console.WriteLine("Введите элементы очереди (для завершения
введите 'end'):");
        string input;
        while ((input = Console.ReadLine()) != "end")
        {
```

```
            Console.Write("Введите приоритет: ");
            int priority;
            while (!int.TryParse(Console.ReadLine(), out priority) ||
priority < 0)
            {
                    Console.WriteLine("Некорректный ввод, попробуйте
еще раз:");
            }
            queue.Enqueue(input, priority);
        }

        Console.WriteLine("Очередь с приоритетами:");

        while (!queue.IsEmpty)
        {
            string item = queue.Dequeue();
            Console.WriteLine(item);
        }
    }
}
```

Далее переделали представленный в примере код (с языка C в язык C#) и реализовали структуру данных «Очередь»:

<u>Описание кода программы 2:</u>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace exampleLaba
```

```
{
    //Класс `Node` представляет узел в очереди и содержит два поля
    public class Node
    {
        //`inf` типа `string`, которое хранит информацию об объекте
        public string inf;
        //`next` типа `Node`, которое указывает на следующий узел в
очереди
        public Node next;
    }
    //Класс `Program` содержит два статических поля:
    public class Program
    {
        //Представляет начало очереди
        public static Node head;
        //Представляет конец очереди
        public static Node last;

        //
        public static void Main(string[] args)
        {
            string menuOption;
            //В цикле выводится меню с опциями для выбора
            //Пользователь вводит выбранную опцию, и в зависимости от
            //этой опции вызывается соответствующий метод или
выводится
            //сообщение об ошибке в случае некорректного выбора
            do
            {
                Console.WriteLine("Выберите опцию:");
                Console.WriteLine("1.Добавить элемент в очередь");
```

```
                Console.WriteLine("2.Посмотреть очередь");
                Console.WriteLine("3.Удалить элемент из очереди");
                Console.WriteLine("4.Выход");

                menuOption = Console.ReadLine();

                switch (menuOption)
                {
                    case "1":
                        Enqueue();
                        break;
                    case "2":
                        review();
                        break;
                    case "3":
                        Dequeue();
                        break;
                    case "4":
                        Console.WriteLine("Программа завершена");
                        break;
                    default:
                        Console.WriteLine("Неверная опция, попробуйте
ещё раз");
                        break;
                }
                Console.WriteLine();
            } while (menuOption != "4");
        }

        //Метод добавляет новый узел в очередь.
```

```
        public static void Enqueue()
        {
            //Создается новый узел `p` с использованием метода `CreateNode()`
            Node p = CreateNode();
            //Если очередь пуста, то новый узел становится началом и концом очереди
            if (head == null && p != null)
            {
                head = p;
                last = p;
            }
            //Если очередь не пуста, то новый узел добавляется в конец очереди, а `last` указывает на него.
            else if (head != null && p != null)
            {
                last.next = p;
                last = p;
            }
            return;
        }

        //Метод `CreateNode` создает новый узел и запрашивает у пользователя данные об объекте.
        public static Node CreateNode()
        {
            Node p = new Node();
            string s;

            Console.WriteLine("Введите название объекта:");
            s = Console.ReadLine();
```

```
            //Если пользователь не ввел название объекта, то
возвращается `null`


            if (string.IsNullOrEmpty(s))

            {

                Console.WriteLine("Запись не была произведена");

                return null;

            }

            //Иначе, узел заполняется данными и возвращается.

            p.inf = s;


            p.next = null;


            return p;

        }


        //Метод `review` выводит содержимое очереди

        public static void review()

        {

            //Переменная `struc` указывает на первый узел в очереди

            Node struc = head;

            //Если очередь пуста, то выводится сообщение об этом

            if (head == null)

            {

                Console.WriteLine("Очередь пуста");

            }

            //Иначе, в цикле выводится название объекта из каждого
узла, пока не достигнется конец очереди.

            int i = 0;

            while (struc != null)

            {
```

```
                Console.WriteLine("{0} - {1}",i,struc.inf);
                i++;
                struc = struc.next;
            }
            return;
        }

        //Метод `Dequeue` удаляет первый узел из очереди.
        public static void Dequeue()
        {
            //Если очередь пуста, выводится сообщение об этом
            if (head == null)
            {
                Console.WriteLine("Очередь пуста");
                return;
            }
            Console.WriteLine("Текущая очередь:");
            review();
            //Пользователь вводит индекс элемента, который нужно
удалить
            Console.WriteLine("Введите индекс элемента, который вы
хотите удалить:");
            int index = Convert.ToInt32(Console.ReadLine());

            //Проверяем, что индекс находится в пределах очереди
            if (index < 0 || index >= Count())
            {
                Console.WriteLine("Неверный индекс");
                return;
            }
```

```
            //Если индекс равен 0, удаляем первый элемент
            if (index == 0)
            {
                head = head.next;
                return;
            }

            //Ищем элемент, предшествующий удаляемому
            Node prev = head;
            for (int i = 1; i < index; i++)
            {
                prev = prev.next;
            }

            //Удаляемый элемент
            Node curr = prev.next;

            //Перестраиваем связи
            prev.next = curr.next;

            //Если удаляемый элемент является последним, обновляем
ссылку на last
            if (curr == last)
            {
                last = prev;
            }

            Console.WriteLine("Элемент удален из очереди");
        }
```

```
        //Метод возвращает количество элементов в очереди
        public static int Count()
        {
            int count = 0;
            Node current = head;
            while (current != null)
            {
                count++;
                current = current.next;
            }
            return count;
        }
    }
}
```

Далее на основе переделанного кода реализовали структуру данных «Стек»

<u>Описание кода программы 3:</u>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laba3Stek
{
    //Класс `Node` представляет узел в стеке и содержит два поля
    public class Node
    {
        //поле `inf`, которое хранит информацию об объекте
        public string inf;
```

```
        //поле `next', которое указывает на следующий узел в стеке
        public Node next;
    }

    public class Program
    {
        //определено статическое поле `top`, которое представляет верхний элемент стека
        public static Node top;

        public static void Main(string[] args)
        {
            string menuOption;
            //цикл выводит меню с опциями для выбора
            //пользователь вводит выбранную опцию, и в зависимости от этой опции
            //вызывается соответствующий метод или выводится сообщение об ошибке в случае некорректного выбора
            do
            {
                Console.WriteLine("Выберите опцию:");
                Console.WriteLine("1.Добавить элемент в стек");
                Console.WriteLine("2.Просмотреть стек");
                Console.WriteLine("3.Удалить элемент из стека");
                Console.WriteLine("4.Выход");

                menuOption = Console.ReadLine();

                switch (menuOption)
                {
                    case "1":
```

```
                        Push();
                        break;
                    case "2":
                        ViewStack();
                        break;
                    case "3":
                        Pop();
                        break;
                    case "4":
                        Console.WriteLine("Программа завершена");
                        break;
                        default:
                        Console.WriteLine("Неверная опция, попробуйте
ещё раз");
                        break;
                }
                Console.WriteLine();
            } while (menuOption != "4");
        }
        //метод добавляет новый узел в стек
        public static void Push()
        {
            //создается новый узел `p` с использованием метода
`CreateNode`
            Node p = CreateNode();
            //Если узел `p` не равен `null`, то узел `p` становится
новой вершиной стека
            //а предыдущая вершина стека становится следующим узлом
для новой вершины
            if (p != null)
            {
                p.next = top;
```

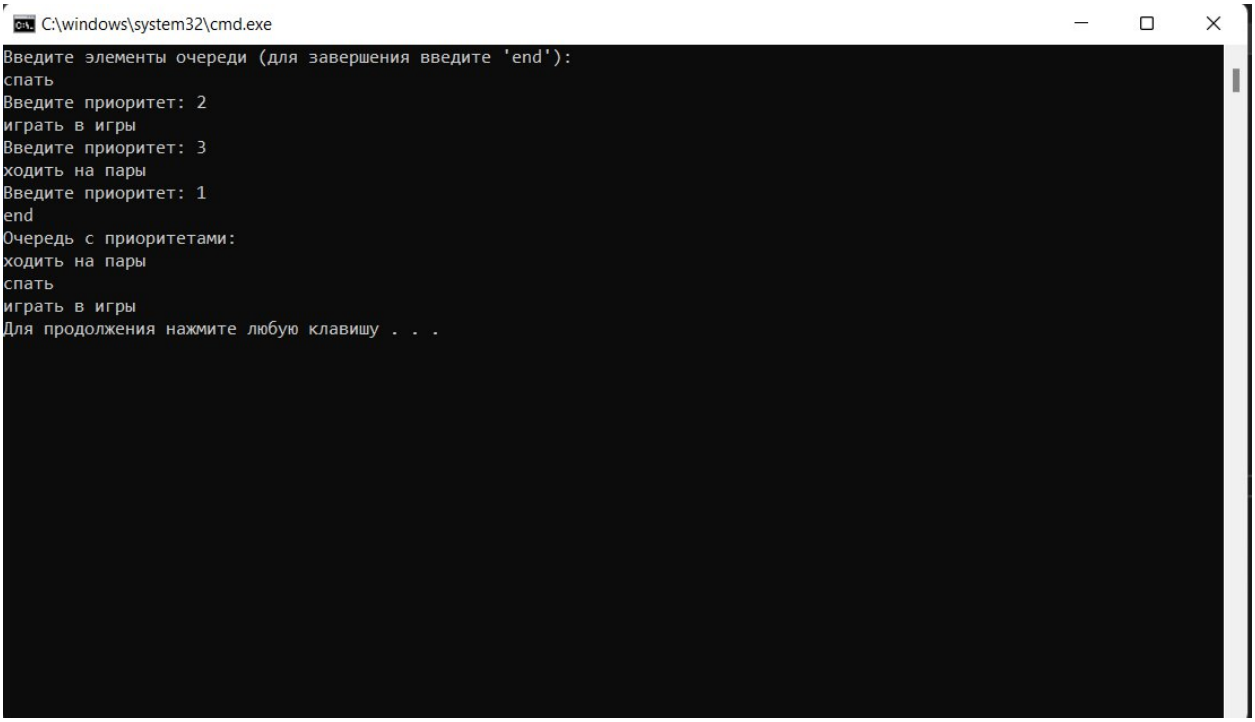
```
                top = p;
            }
            return;
        }

        //Метод создает новый узел и запрашивает у пользователя данные об объекте
        public static Node CreateNode()
        {
            Node p = new Node();
            string s;

            Console.WriteLine("Введите название объекта:");
            s = Console.ReadLine();
            //если пользователь не ввел название объекта, то возвращается `null`
            if (string.IsNullOrEmpty(s))
            {
                Console.WriteLine("Запись не была произведена");
                return null;
            }
            //иначе узел заполняется данными и возвращается
            p.inf = s;

            p.next = null;

            return p;
        }

        //метод выводит содержимое стека
        public static void ViewStack()
```

```
        {
            //переменная `struc` указывает на вершину стека
            Node struc = top;
            //если стек пуст, то выводится соответствующее сообщение
            if (top == null)
            {
                Console.WriteLine("Стек пуст");
            }

            //иначе, в цикле выводится название объекта из каждого
узла, пока не достигнется конец стека
            while (struc != null)
            {
                Console.WriteLine("Имя - {0}", struc.inf);
                struc = struc.next;
            }
            return;
        }

        //метод удаляет верхний элемент из стека
        public static void Pop()
        {
            //если стек пуст, выводится сообщение об этом
            if (top == null)
            {
                Console.WriteLine("Стек пуст");
                return;
            }
            //иначе, верхний узел удаляется из стека, и информация об
этом выводится на экран
            Node poppedNode = top;
```

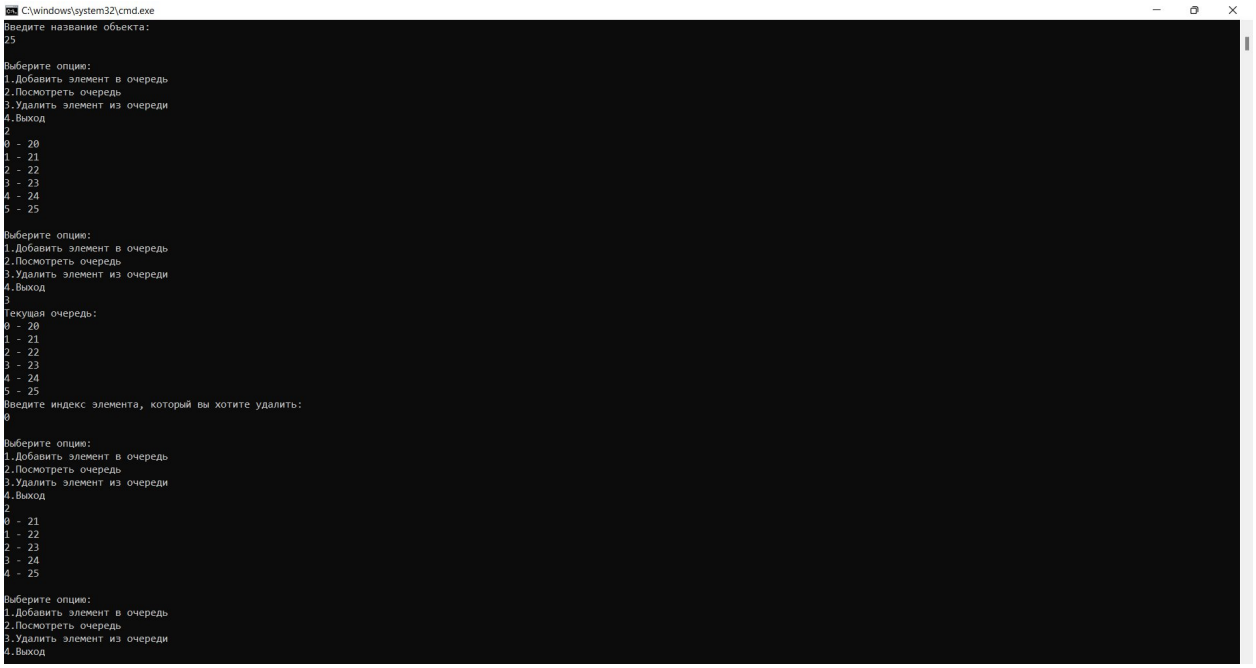
```
            top = top.next;
            poppedNode.next = null;

            Console.WriteLine("Элемент {0} удален из стека",
poppedNode.inf);
        }

        //метод проверяет, содержится ли объект с заданным именем в
стеке
        public static bool Contains(string name)
        {
            //переменная `struc` указывает на вершину стека
            Node struc = top;
            //если стек пуст, выводится сообщение об этом
            if (top == null)
            {
                Console.WriteLine("Стек пуст");
            }
            //иначе, в цикле проверяется каждый узел на соответствие
имени
            while (struc != null)
            {
                //если совпадение найдено, возвращается `true`
                if (name == struc.inf)
                {
                    return true;
                }
                struc = struc.next;
            }
            //если цикл завершается без совпадений, возвращается
`false`
            return false;
```

```
            }
        }
}
```

Результат работы программы 1:



Результат работы программы 2:



Результат работы программы 3:

Вывод: научились реализовывать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта, также реализовывать такие структуры данных, как «Очередь» и «Стек».