

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.

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;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _3laba
{
    public class PriorityQueue<T>
    {
        //объявление приватного поля `queue`, которое будет использоваться
        //для хранения элементов и их приоритетов в виде списка кортежей.
        private List<Tuple<T, int>> queue;
        //конструктор класса, который инициализирует список `queue`.
        public PriorityQueue()
        {
            queue = new List<Tuple<T, int>>();
        }

        //метод, который добавляет элемент и его приоритет в очередь
        public void Enqueue(T item, int priority)
        {
            //Кортеж добавляется в список `queue`.
            queue.Add(new Tuple<T, int>(item, priority));
            //Список сортируется в порядке убывания приоритетов элементов.
            queue.Sort((x, y) => y.Item2.CompareTo(x.Item2));
        }

        //метод извлекает элемент с наивысшим приоритетом из очереди
```

```

    public T Dequeue()
    {
        //Проверяется, что список `queue` не пустой, иначе вызывается
        //исключение.
        if (queue.Count == 0)
            throw new InvalidOperationException("Queue is empty");
        //Извлекается элемент с наивысшим приоритетом из списка
        `queue`.
        T item = queue[0].Item1;
        //Элемент удаляется из списка `queue`.
        queue.RemoveAt(0);
        //Извлеченный элемент возвращается.
        return item;
    }
    //свойство `Count`, которое возвращает количество элементов в
    очереди.
    public int Count
    {
        get { return queue.Count; }
    }
    //свойство `IsEmpty`, которое возвращает `true`, если очередь
    пуста, иначе `false`.
    public bool IsEmpty
    {
        get { return queue.Count == 0; }
    }

    public void Clear()
    {
        queue.Clear();
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        //создание экземпляра класса
        PriorityQueue<string> queue = new PriorityQueue<string>();

        //вызов метода `Enqueue` для добавления элемента "Item 1"
        с приоритетом 3 в очередь.
        queue.Enqueue("Item 1", 3);
        //вызов метода `Enqueue` для добавления элемента "Item 2"
        с приоритетом 1 в очередь.
        queue.Enqueue("Item 2", 1);
        //вызов метода `Enqueue` для добавления элемента "Item 3"
        с приоритетом 2 в очередь.
        queue.Enqueue("Item 3", 2);

        //цикл будет выполняться, пока очередь не пуста
    }
}

```

```

        while (!queue.IsEmpty)
        {
            //вызов метода `Dequeue` для извлечения элемента с наивысшим
            приоритетом из очереди и присваивание его переменной `item`
            string item = queue.Dequeue();
            Console.WriteLine(item);
        }
    }
}

```

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

Описание кода программы 2:

```

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.
    while (struc != null)
    {
        Console.WriteLine("Имя - {0}", struc.inf);
        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;
    }
}

```



```
        //Иначе, первый узел удаляется из очереди, и информация об  
этом выводится на экран.
```

```
        Node dequeuedNode = head;
```

```
        head = head.next;
```

```
        dequeuedNode.next = null;
```

```
        Console.WriteLine("Элемент {0} удален из очереди",  
dequeuedNode.inf);
```

```
    }
```

```
    //Метод `Contains` проверяет, содержится ли объект с заданным  
именем в очереди
```

```
    public static bool Contains(string name)
```

```
    {
```

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

```
        Node struc = head;
```

```
        // Если очередь пуста, выводится сообщение об этом
```

```
        if (head == null)
```

```
        {
```

```
            Console.WriteLine("Очередь пуста");
```

```
        }
```

```
        //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;
    }
}
}

```

Далее на основе переделанного кода реализовали структуру данных «Стек»

Описание кода программы 3:

```

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);
}

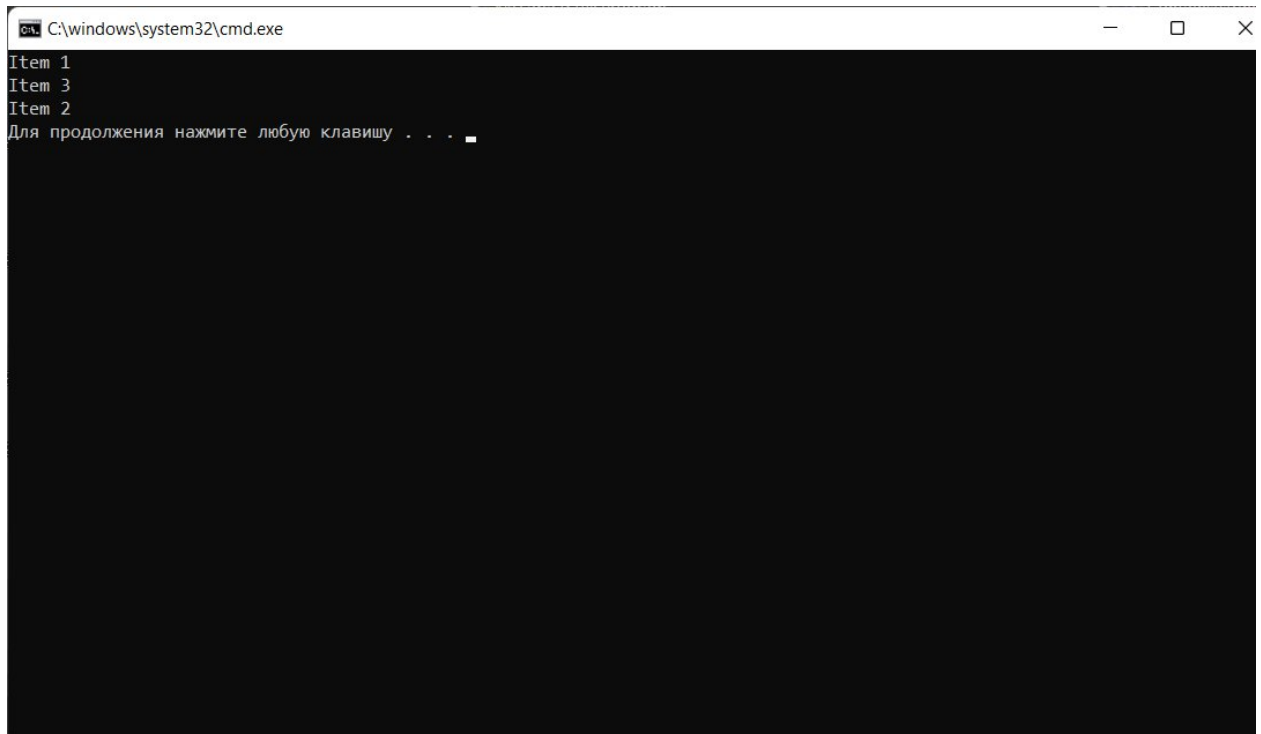
```

```

//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:



```
C:\windows\system32\cmd.exe
Item 1
Item 3
Item 2
Для продолжения нажмите любую клавишу . . .
```

Результат работы программы 2:



```
C:\windows\system32\cmd.exe
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
1
Введите название объекта:
Дом
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
2
Дом - Дом
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
3
Элемент Дом удален из очереди
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
1
Введите название объекта:
Дом
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
1
Введите название объекта:
крыша
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
1
```



```
C:\windows\system32\cmd.exe
3
Элемент дом удален из очереди
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
3
Элемент крыша удален из очереди
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
3
Очередь пуста
Выберите опцию:
1. Добавить элемент в очередь
2. Посмотреть очередь
3. Удалить элемент из очереди
4. Выход
```

Результат работы программы 3:

```
C:\windows\system32\cmd.exe
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
1
Введите название объекта:
дом
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
1
Введите название объекта:
сарай
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
2
Имя - сарай
Имя - дом
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
3
Элемент сарай удален из стека
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
3
Элемент дом удален из стека
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
2
Стек пуст
Выберите опцию:
1. Добавить элемент в стек
2. Посмотреть стек
3. Удалить элемент из стека
4. Выход
```

Вывод: научились реализовывать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом), также реализовывать такие структуры данных, как «Очередь» и «Стек»