

Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

**ОТЧЕТ**  
по лабораторной работе №3  
по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Динамические списки»

Выполнили:

студенты группы 24ВВВ3

Агапов И.А.

Любченко В.К.

Приняли:

к.т.н., доцент Юрова О.В.

к.т.н., Деев М.В.

Пенза 2025

**Цель работы** – Освоить принципы работы с динамическими структурами данных на примере односвязных списков, включая их создание, манипулирование элементами и организацию памяти.

### Лабораторное задание:

#### Задание

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*.

### Задание 1.

### Код программы

C#

```
using System.Collections.Generic;
using System.Security.Cryptography;

namespace Laba3_ALVT
{
    public class Node
    {
        public string Data { get; set; }           // Полезная информация
        public int Priority { get; set; }          // Приоритет
        public Node Next { get; set; }            // Ссылка на следующий узел

        public Node(string data, int priority)
        {
            Data = data;
            Priority = priority;
            Next = null;
        }
    }

    public class Prioretet
    {
        private Node head;

        public Prioretet()
        {
            head = null;
        }

        public void Enqueue(string data, int prioretet)
        {
            Node newNode = new Node(data, prioretet);
            if (head == null)
            {
                head = newNode;
                Console.WriteLine($"Добавлен довый элемент: {data}");
            }

            if (prioretet > head.Priority)
```

```

    {
        newNode.Next = head;
        head = newNode;
        return;
    }

    Node pervious = null;
    Node current = head;

    while (current != null && current.Priority >= prioretet)
    {
        pervious = current;
        current = current.Next;
    }
    pervious.Next = newNode;
    newNode.Next = current;
}

public void Dispaly()
{
    if (head == null)
    {
        Console.WriteLine("Очередь пуста!");
    }
    Node current = head;
    int position = 1;
    Console.WriteLine("\n--- Содержимое очереди ---");
    while (current != null)
    {
        Console.WriteLine($"{position}. Данные: '{current.Data}', Приоритет: {current.Priority}");
        current = current.Next;
        position++;
    }
}

class Program()
{
    static void AddElement(Prioretet queue)
    {
        Console.Write("Введите данные: ");
        string data = Console.ReadLine();
        Console.Write("Введите приоритет (число): ");
        if (int.TryParse(Console.ReadLine(), out int priority))
        {
            queue.Enqueue(data, priority);
            Console.WriteLine("Элемент добавлен!");
        }
        else
        {
            Console.WriteLine("Ошибка: приоритет должен быть числом!");
        }
    }

    static void Main()
    {
        //Prioretet testQueue = new Prioretet();
        //testQueue.Enqueue("Обычная задача", 3);
        //testQueue.Enqueue("Аварийная", 10);
        //testQueue.Enqueue("Срочная", 7);
    }
}

```

```

//testQueue.Enqueue("Не важно", 1);
//testQueue.Dispaly();
//Console.WriteLine();

Prioretet queue = new Prioretet();
bool runing = true;

Console.WriteLine("=== ПРИОРИТЕТНАЯ ОЧЕРЕДЬ ===");
while (true)
{
    Console.WriteLine("\n1. Добавить элемент");
    Console.WriteLine("2. Просмотреть очередь");
    Console.WriteLine("3. Выйти");
    Console.Write("Выберите действие: ");

    string choise = Console.ReadLine();

    switch (choise)
    {
        case "1":
            AddElement(queue);
            break;
        case "2":
            queue.Dispaly();
            break;
        case "3":
            runing = false;
            break;
        default:
            Console.WriteLine("Неверный вобор!");
            break;
    }
}
}
}
}
}

```

**Результат работы программы**

```

=== ПРИОРИТЕТНАЯ ОЧЕРЕДЬ ===

1. Добавить элемент
2. Просмотреть очередь
3. Выйти
Выберите действие: 1
Введите данные: пгщшшрврпы
Введите приоритет (число): 1
Добавлен довший элемент: пгщшшрврпы
Элемент добавлен!

1. Добавить элемент
2. Просмотреть очередь
3. Выйти
Выберите действие: 1
Введите данные: фкфщаылдмлдыв
Введите приоритет (число): 2
Элемент добавлен!

1. Добавить элемент
2. Просмотреть очередь
3. Выйти
Выберите действие: 2

--- Содержимое очереди ---
1. Данные: 'фкфщаылдмлдыв', Приоритет: 2
2. Данные: 'пгщшшрврпы', Приоритет: 1

1. Добавить элемент
2. Просмотреть очередь

```

Рисунок 1 - Результат работы программы 1

## Задание 2.

### Код программы

C#

```

using System.Diagnostics.Contracts;

namespace Laba3_ALVT
{
    public class Node
    {
        public string Data { get; set; } // Полезная информация
        public int Priority { get; set; } // Приоритет
        public Node Next { get; set; } // Ссылка на следующий узел

        public Node(string data)
        {
            Data = data;
            Next = null;
        }
    }

    class Queue
    {
        private Node head;
        private Node tail;

        public Queue()
        {
            head = null;
            tail = null;
        }
    }
}

```

```

}

public void Enqueue(string data)
{
    Node newNode = new Node(data);
    if (head == null)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        tail.Next = newNode;
        tail = newNode;
    }
}

public string DeQueue()
{
    if (head == null)
    {
        Console.WriteLine("Очередь пуста");
        return null;
    }
    string data = head.Data;
    head = head.Next;

    if (head == null)
    {
        tail = null;
    }

    Console.WriteLine($"Извлеченный элемент {data}");
    return data;
}

public string First()
{
    if (head == null)
    {
        Console.WriteLine("Очередь пуста");
        return null;
    }

    string data = head.Data;
    Console.WriteLine($"Первый элемент: {data}");
    return data;
}

public void Display()
{
    if (head == null)
    {
        Console.WriteLine("Очередь пуста");
    }

    Node current = head;
    int plase = 1;

    Console.WriteLine("\n--- Содержимое очереди ---");
}

```

```

        while (current != null)
        {
            Console.WriteLine($"Номер: {plase}, Элемент: {current.Data}");
            current = current.Next;
            plase++;
        }
    }
}

class Prog
{
    static void Main()
    {
        Queue queue = new Queue();

        queue.Enqueue("Первый");
        queue.Enqueue("Второй");
        queue.Enqueue("Третий");

        queue.Display();
        // Вывод:
        // 1. Первый
        // 2. Второй
        // 3. Третий
        Console.WriteLine();
        Console.WriteLine(queue.First());

        queue.DeQueue(); // Уйдет "Первый"
        queue.DeQueue(); // Уйдет "Второй"

        queue.Display();
        // Вывод:
        // 1. Третий

        queue.DeQueue(); // Уйдет "Третий"
        queue.DeQueue(); // Очередь пуста!

    }
}

```

**Результат работы программы**

```
--- Содержимое очереди ---  
Номер: 1, Эллемент: Первый  
Номер: 2, Эллемент: Второй  
Номер: 3, Эллемент: Третий  
  
Первый элемент: Первый  
Первый  
Извлеченный элемент Первый  
Извлеченный элемент Второй  
  
--- Содержимое очереди ---  
Номер: 1, Эллемент: Третий  
Извлеченный элемент Третий  
Очередь пуста
```

Рисунок 2 - Результат работы программы 2

### Задание 3.

#### Код программы

C#

```
using System.Globalization;  
  
namespace Laba3_ALVT  
{  
    public class Node  
    {  
        public string Data { get; set; } // Полезная информация  
        public Node Next { get; set; } // Ссылка на следующий узел  
  
        public Node(string data)  
        {  
            Data = data;  
            Next = null;  
        }  
    }  
  
    class Steck  
    {  
  
        private Node top;  
        private Node tail;  
  
        public Steck()  
        {  
            top = null;  
            tail = null;  
        }  
  
        public string Peek()  
        {  
  
            if (top == null)  
            {  
                Console.WriteLine("Стек пуст");  
            }  
        }  
    }  
}
```



```

        return null;
    }
    else
    {
        Console.WriteLine($"Первый элемент: {top.Data}");
        return top.Data;
    }
}

public void Add(string Data)
{
    Node NewNode = new Node(Data);
    NewNode.Next = top;
    top = NewNode;
}

public string Pop()
{
    if (top == null)
    {
        Console.WriteLine("Стек пуст");
        return null;
    }
    else
    {
        string data = top.Data;
        top = top.Next;
        Console.WriteLine($"Удален элемент: {data}");
        return data;
    }
}

}
public void Display()
{
    if (top == null)
    {
        Console.WriteLine("Стек пуст");
        return;
    }

    Node current = top;
    while (current != null)
    {
        Console.WriteLine(current.Data);
        current = current.Next;
    }
}
}
class Prog
{
    static void Main()
    {
        Steck st = new Steck();
        st.Add("Первый");
        st.Add("Второй");
        st.Add("Третий");
        st.Add("Четвертый");

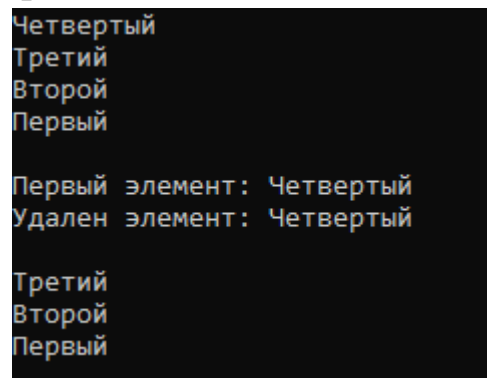
        st.Display();
        Console.WriteLine();
        st.Peek();

        st.Pop();
        Console.WriteLine();
    }
}

```

```
        st.Display();  
  
        Console.ReadKey();  
    }  
}
```

### Результат работы программы



```
Четвертый  
Третий  
Второй  
Первый  
  
Первый элемент: Четвертый  
Удален элемент: Четвертый  
  
Третий  
Второй  
Первый
```

Рисунок 3 - Результат работы программы 3

**Вывод:** В ходе выполнения лабораторной работы были успешно освоены принципы работы с динамическими структурами данных на примере односвязных списков.