# Московский авиационный институт (национальный исследовательский университет)

## Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу ООП: Конкретизация. Обобщенные классы.

Работу выполі	нила:			
М8O-209Б-19	Офицерова	а Т.И		
Группа	ФИО	Подпись	Вариант	
Руководитель	<b>:</b> Подпись:	_/Кузнецова С.В	3./	

Дата: 18 октября 2020

#### Текст программы

```
using System;
namespace lab8
    using System;
    namespace lab0
        class Stack<T> //стек обобщенный
            private T[] items;
            private int count;
            private const int n = 10;
            public Stack()//создание стека с 10 местами
                items = new T[n];
            }
            public Stack(int length)//создать стек заданной длины
                items = new T[length];
            }
            private void Resize(int length)//изменить размер стека
                T[] tmp = new T[length];
                int min = Math.Min(items.Length, length);
                for (int i = 0; i < min; i++)</pre>
                    tmp[i] = items[i];
                items = tmp;
            }
            public bool IsEmpty//проверка на пустоту
            {
                get { return count == 0; }
            }
            public int Size//вернуть размер
                get { return items.Length; }
            public int Count()//вернуть количество элементов
                return count;
            }
            public void Push(T item)//вставка
                if (count == items.Length)
                    Resize(count + 1);
                items[count++] = item;
            }
            public T Pop()//вынуть и узнать элемент
                if (IsEmpty)
                    throw new InvalidOperationException("Cτeκ πycτ");
                T item = items[--count];
                items[count] = default(T);
                return item;
            }
            public T Check()//узнать последний элемент
                return items[count - 1];
            public void Print()//вывод стека
```

```
for (int i = 0; i < count; i++)
            {
                Console.Write("{0} ", items[i]);
            Console.WriteLine();
        }
    }
    class Program
        static void Main(string[] args)
            Stack<int> stack1 = new Stack<int>();//стек с целыми числами
            Random rand = new Random();
            for (int i = 0; i < stack1.Size; i++)</pre>
            {
                stack1.Push(rand.Next(0, 100));
            }
            stack1.Print();
            stack1.Push(rand.Next(0, 100));
            stack1.Print();
            Console.WriteLine("Количество элементов в стеке: {0}", stack1.Count());
            Console.WriteLine("Вынутый элемент стека: {0}", stack1.Pop());
            stack1.Print();
            Console.WriteLine("Последний элемент в стеке: {0}\n", stack1.Check());
            Stack<string> stack2 = new Stack<string>(5);//стек со строками
            for (int i = 0; i < stack2.Size; i++)</pre>
                stack2.Push(rand.Next(0, 10).ToString()+"str");
            }
            stack2.Print();
            stack2.Push(rand.Next(0, 10).ToString() + "str");
            stack2.Print();
            Console.WriteLine("Количество элементов в стеке: {0}", stack2.Count());
            Console.WriteLine("Вынутый элемент стека: {0}", stack2.Pop());
            stack2.Print();
            Console.WriteLine("Последний элемент в стеке: {0}", stack2.Check());
        }
    }
}
```

### Результат работы

```
Консоль отладки Microsoft Visual Studio — □

55 56 75 87 84 22 75 6 15 81

55 56 75 87 84 22 75 6 15 81 92

Количество элементов в стеке: 11

Вынутый элемент стека: 92

55 56 75 87 84 22 75 6 15 81

Последний элемент в стеке: 81

9str 6str 3str 9str 7str

9str 6str 3str 9str 7str 2str

Количество элементов в стеке: 6

Вынутый элемент стека: 2str

9str 6str 3str 9str 7str
```

#### Примеры

Классификация позвоночных, очередь любых объектов

#### Вывод

Конкретизация помогает сделать код более универсальным и полиморфным. Обобщения позволяют не писать один и тот же код для различных типов данных, а использовать один код, конкретизируя тип при создании объекта.