Московский авиационный институт
(национальный исследовательский университет)

**Институт информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

# Лабораторная работа №4 и 5 по курсу ООП:

# Комбинирование, комбинирование через общих предков.

Работу выполнила:
М8О-209Б-19 Офицерова Т.И. _____ _____
Группа ФИО Подпись Вариант

Руководитель: _____/Кузнецова С.В./
Подпись:

Дата: 2020

## Комбинирование

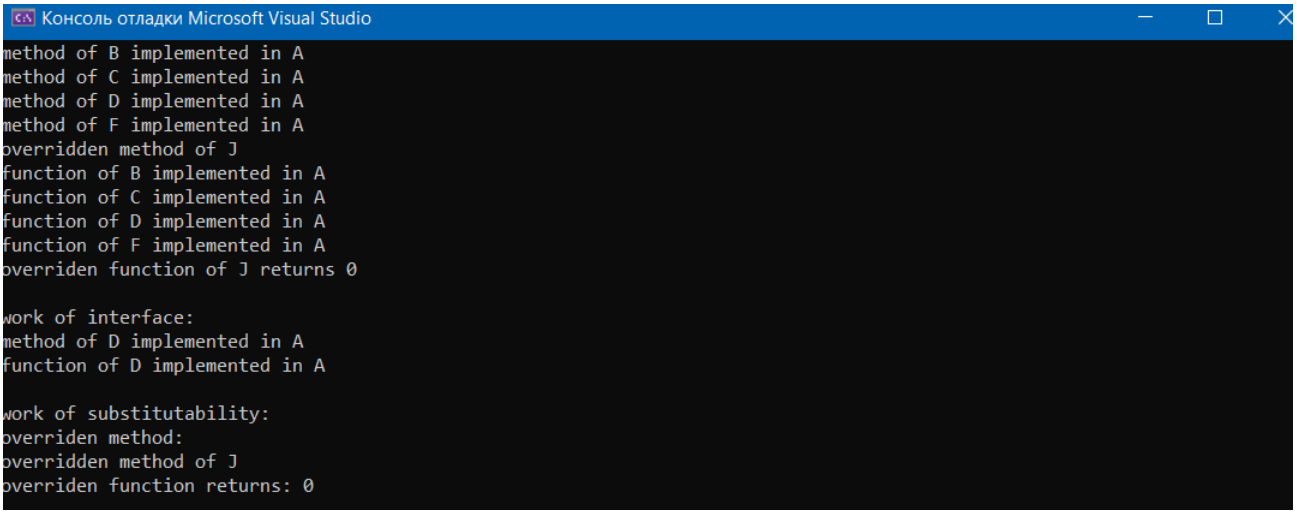
### Текст программы

```
using System;

namespace lab5
{
    class Program
    {
        public interface F
        {
            public void MethodF();
            public string FunctionF();
        }
        public interface D
        {
            public void MethodD();
            public string FunctionD();
        }
        public class J
        {
            private int number;
            public virtual void MethodJ()
            {
                Console.WriteLine("it's method of J");
            }
            public virtual int FunctionJ()
            {
                return this.number;
            }
        }
        public interface B : D
        {
            public void MethodB();
            public string FunctionB();
        }
        public interface C : F
        {
            public void MethodC();
            public string FunctionC();
        }
        public class A : J, B, C
        {
            public string FunctionB()
            {
                return "function of B implemented in A";
            }
            public string FunctionC()
            {
                return "function of C implemented in A";
            }
            public string FunctionD()
            {
                return "function of D implemented in A";
            }
            public string FunctionF()
            {
                return "function of F implemented in A";
            }
            public void MethodB()
            {
                Console.WriteLine("method of B implemented in A");
            }
            public void MethodC()
```

```
            {
                Console.WriteLine("method of C implemented in A");
            }
            public void MethodD()
            {
                Console.WriteLine("method of D implemented in A");
            }
            public void MethodF()
            {
                Console.WriteLine("method of F implemented in A");
            }
            public override void MethodJ()
            {
                Console.WriteLine("overridden method of J");
            }
            public override int FunctionJ()
            {
                return 0;
            }
        }
        static void Main(string[] args)
        {
            A a = new A();//создание объекта A
            a.MethodB();//вызов реализованных и наследованных методов
            a.MethodC();
            a.MethodD();
            a.MethodF();
            a.MethodJ();
            Console.WriteLine(a.FunctionB());
            Console.WriteLine(a.FunctionC());
            Console.WriteLine(a.FunctionD());
            Console.WriteLine(a.FunctionF());
            Console.WriteLine("overriden function of J returns {0}", a.FunctionJ());
            Console.WriteLine("\nwork of interface:");//работа через интерфейс
            D d = new A();
            d.MethodD();
            Console.WriteLine(d.FunctionD());
            Console.WriteLine("\nwork of substitutability:");//проверка подстановки
            J j = new A();
            Console.WriteLine("overriden method:");
            j.MethodJ();
            Console.WriteLine("overriden function returns: {0}", j.FunctionJ());        }
    }
}
```

**Результат работы**



### Примеры

Мультитул, мультиварка

## Комбинирование через общих предков

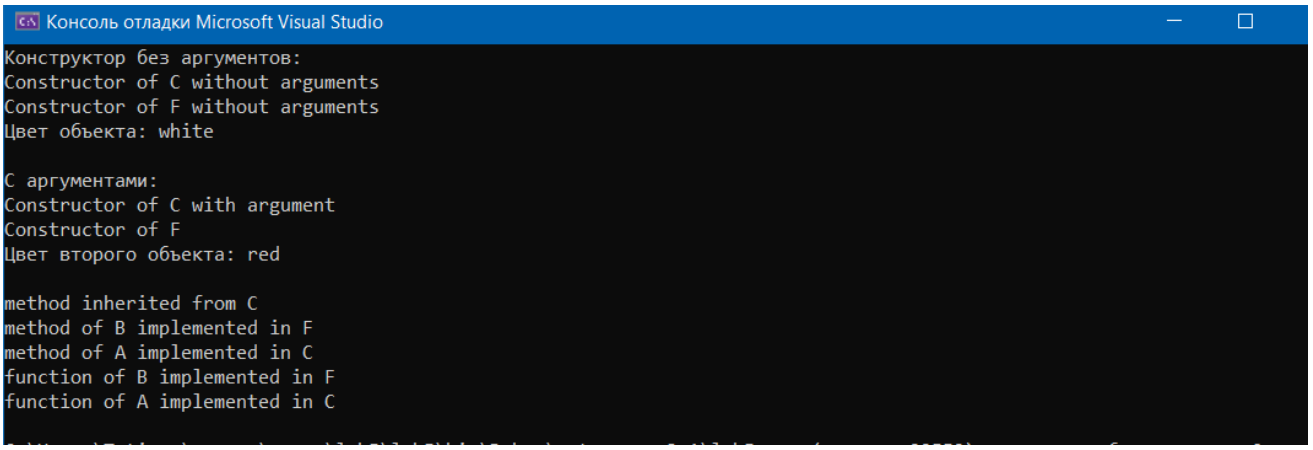
### Текст программы

```
using System;
using System.Collections.Generic;
using System.Text;

namespace lab5
{
    public interface A
    {
        public void MethodA();
        public string FunctionA();
    }
    public interface B : A
    {
        public void MethodB();
        public string FunctionB();
    }
    public class C : A
    {
        protected string color;
        public string FunctionA()
        {
            return "function of A implemented in C";
        }
        public void MethodA()
        {
            Console.WriteLine("method of A implemented in C");
        }
        public string GetColor()//узнать цвет
        {
            return this.color;
        }
        public C()//конструктор без аргументов
        {
            Console.WriteLine("Constructor of C without arguments");
            this.color = "black";
        }
        public C(string color)//конструктор с аргументом
        {
            Console.WriteLine("Constructor of C with argument");
            this.color = color;
        }
        public void MethodC()
        {
            Console.WriteLine("method inherited from C");
        }
    }
    public class F : C, B
    {
        public F()//конструктор без аргументов
        {
            Console.WriteLine("Constructor of F without arguments");
            this.color = "white";
        }
        public F(string color) : base(color) //конструктор с аргументами
        {
            Console.WriteLine("Constructor of F");
        }
        public string FunctionB()
```

```
        {
            return "function of B implemented in F";
        }
        public void MethodB()
        {
            Console.WriteLine("method of B implemented in F");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Конструктор без аргументов:");//проверка работы
конструкторов
            F f1 = new F();
            Console.WriteLine("Цвет объекта: {0}", f1.GetColor());
            Console.WriteLine("\nC аргументами:");
            F f2 = new F("red");
            Console.WriteLine("Цвет второго объекта: {0}\n", f2.GetColor());
            f2.MethodC();//вызов реализованных методов
            f2.MethodB();
            f2.MethodA();
            Console.WriteLine(f2.FunctionB());//вызов реализованных функций
            Console.WriteLine(f2.FunctionA());
        }
    }
}
```

**Результат работы**




**Примеры**

Диван-кровать, мул, апельсин (гибрид мандарина и помело)

**Вывод**

Комбинирование или множественное наследование позволяет наследовать поведение сразу нескольких видов объектов. Но так как в C# нет множественного наследования классов, оно реализуется с помощью использования интерфейсов.