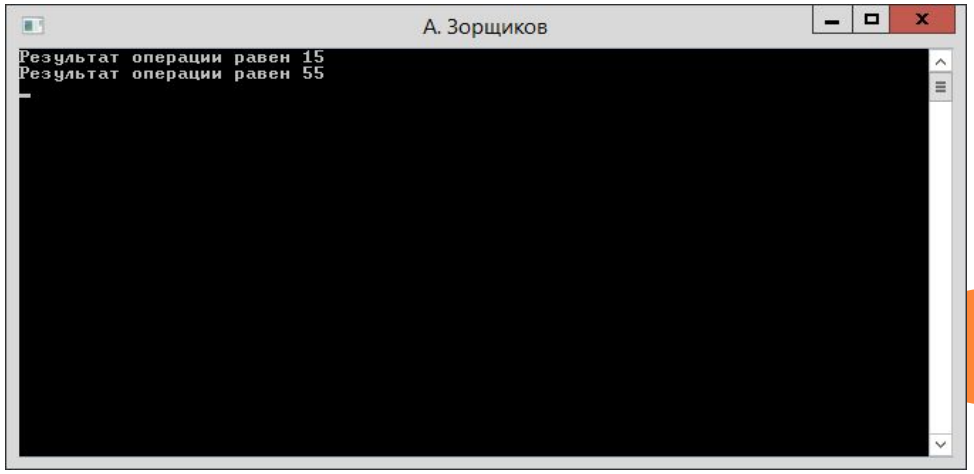
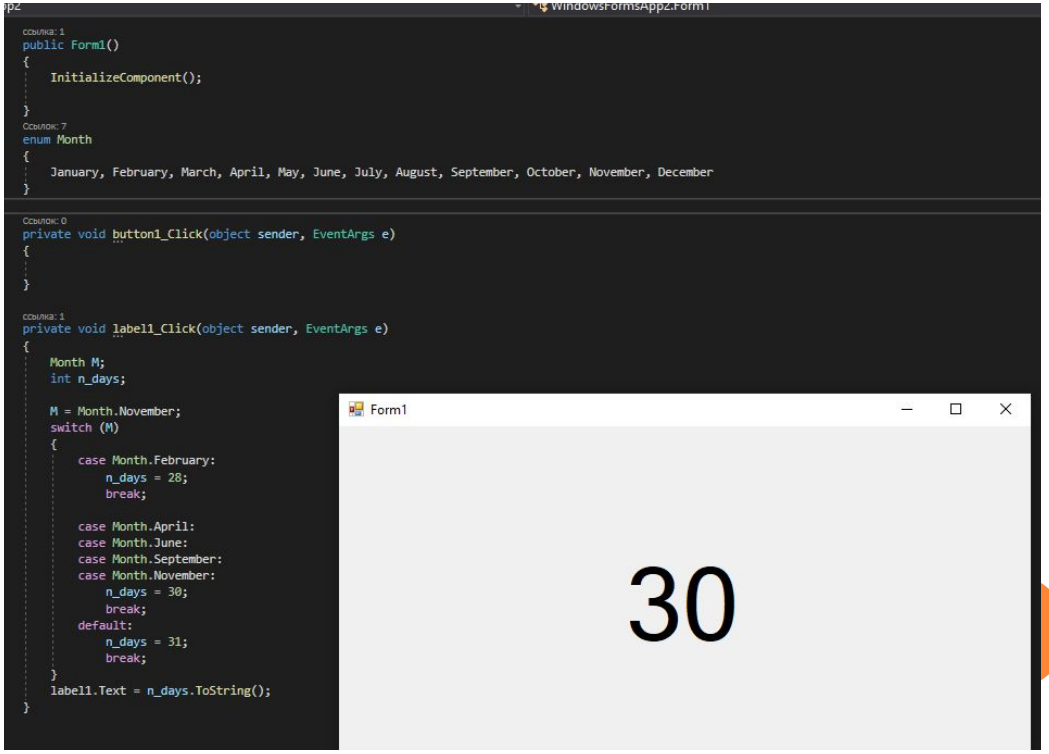
Практическая работа. Пример №1

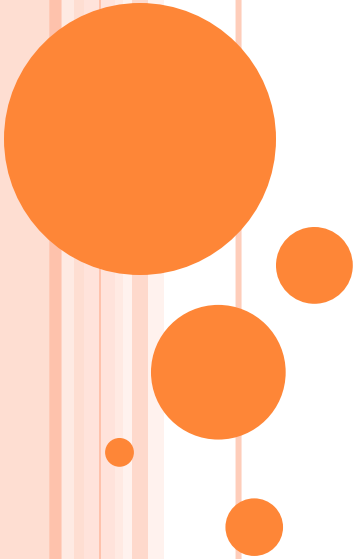
```
…..
staticvoid Main(string[] args)
        {
Console.Title = "А. Зорщиков";
MathOp(10, 5, Operation.Add);
MathOp(11, 5, Operation.Multiply);
Console.ReadLine();
 }
……
```



Практическая работа. Пример №2

- Протестируйте пример программы, в котором приводится использование перечисления Month для определения количества дней в месяце.





Пользовательские типы данных

Структуры

Что представляет собой структура?

Структура, которая подобна классу, но относится к типу значения (value type), а не к ссылочному типу данных (reference type).

Структуры отличаются от классов:

как они сохраняются в памяти и как к ним осуществляется доступ *(классы — это ссылочные типы, размещаемые в куче, структуры — типы значений, размещаемые в стеке)*;

свойствами (например, структуры не поддерживают наследование).

- Как и у классов, у каждой структуры имеются свои члены: методы, поля, свойства, операторные методы и события.
- В структурах допускается также определять конструкторы, но не деструкторы. В то же время для структуры нельзя определить конструктор, используемый по умолчанию (т.е. конструктор без параметров). Потому, что конструктор, вызываемый по умолчанию, определяется для всех структур автоматически и не подлежит изменению. Такой конструктор инициализирует поля структуры значениями, задаваемыми по умолчанию.
- Структуры не поддерживают наследование, соответственно их члены нельзя указывать как abstract, virtual или protected.

- Объект структуры может быть создан с помощью **оператора new** таким же образом, как и объект класса, но в этом нет необходимости. Оператор new вызывает конструктор, используемый по умолчанию.
- Если оператор new не используется, объект все равно создается, хотя и не инициализируется. В этом случае инициализация членов структуры выполняется вручную.

Назначение структур

- Часто, при написании программ, которые содержат разнообразные данные, возникает необходимость группирования этих данных по некоторому критерию.
- Такое группирование улучшает наглядность программного кода, которое в свою очередь приводит к уменьшению ошибок и повышения производительности работы программиста.
- Доступ к сгруппированным данным упрощает использование имен в программе. Данные группируются соответственно проблемной области, для которой разрабатывается программа.
- В языке программирования C#, с целью удобного группирования данных, используются структуры.
- Использование структуры в программе происходит в 2 этапа:

1. объявление типа структуры;
2. объявление структурной переменной.

Общая форма объявления типа структуры:

*При объявлении используется ключевое слово **struct** вместо class*

struct *имя_типа_структуры* : *интерфейсы*

{

// объявление членов и методов структуры

}

- *имя_типа_структуры* – название структурного типа на основе которого будут объявляться объекты (переменные, экземпляры структуры);
- *интерфейсы* – список интерфейсов, методы которых нужно реализовать в теле структуры.

Пример объявления структуры, которая описывает запись в телефонном справочнике:

```
struct Telephone
{
    public string number; // номер телефона
    public string name; // имя абонента
    public string surname; // фамилия абонента
    public string address; // адрес
    public int code; // почтовый код
}
```

Пример объявления, инициализации и использования структурной переменной типа Telephone

```
// Объявление структурной переменной с именем T1 типа Telephone
Telephone T1;
// заполнение полей структурной переменной T1
T1.name = “Иванов";
T1.surname = «Иван";
T1.number = «77058085423";
T1.code = 640712402120;
T1.address = «Уральск";
```

Тип доступа к полям структурной переменной

- По умолчанию, всем членам структуры устанавливается модификатор доступа **private**. Это означает, что при объявлении переменной типа Структура, невозможно будет обратиться к полям структуры непосредственно.
- Для доступа к полям структуры используются следующие модификаторы доступа (private, public).
- Структуры не поддерживают наследование, поэтому их члены нельзя указывать как abstract, virtual или protected.