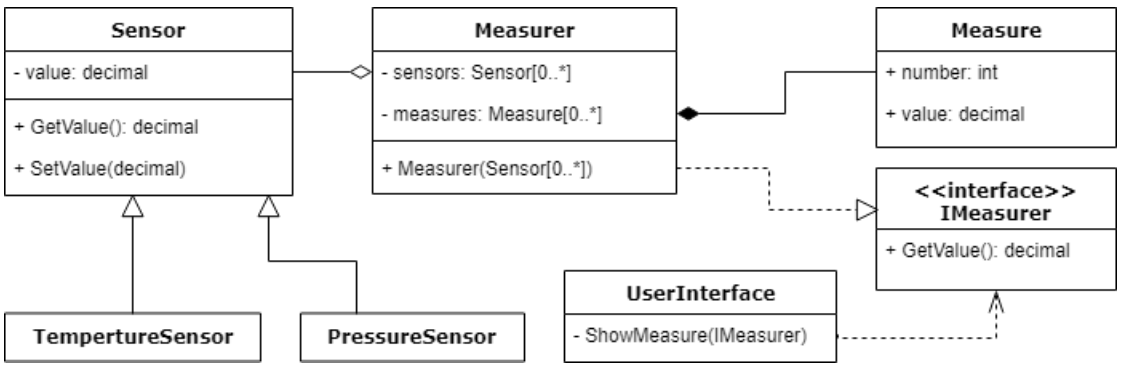
ДЕНЬ 4E. Спецификации

4E.1. Создать декларацию классов C# согласно заданной UML-диаграмме классов

Создать код декларации классов (без их реализации) на языке C# по двум заданным UML-диаграммам классов

Пример

Диаграмма классов:



Код декларации классов:

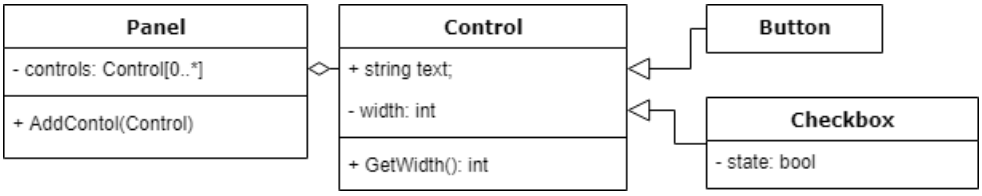
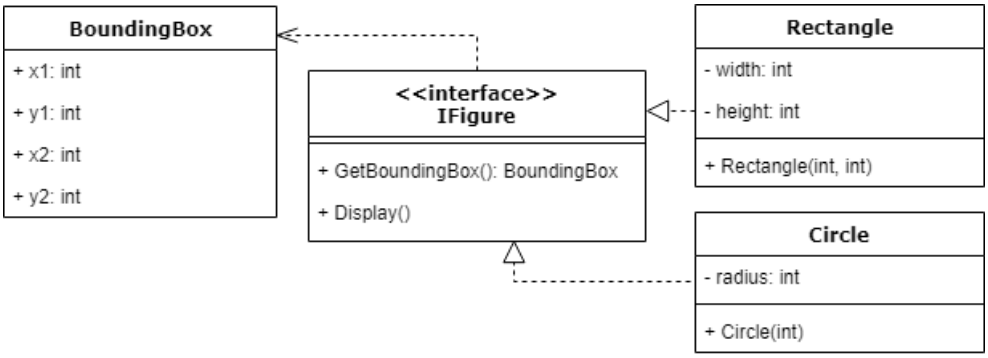
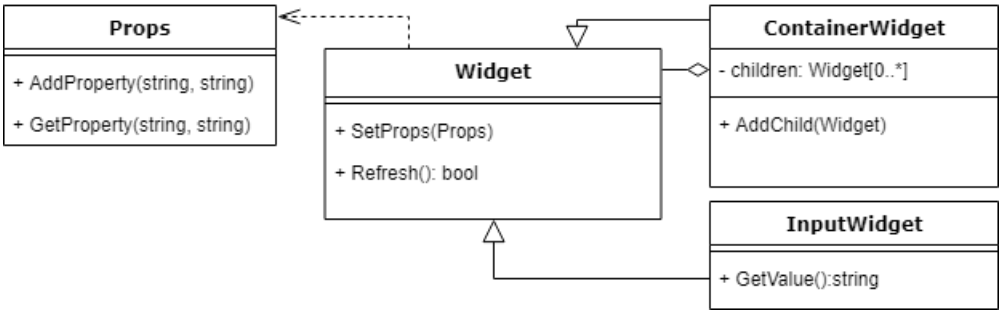
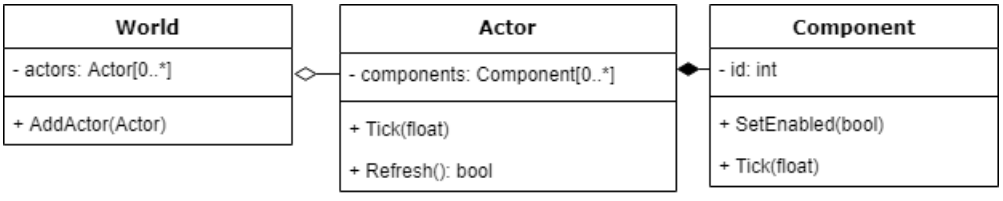
```
public class Sensor
{
    private decimal value;

    public decimal GetValue()
    {
        throw new NotImplementedException();
    }

    public void SetValue(decimal val)
    {
        throw new NotImplementedException();
    }
}

public class TempertureSensor: Sensor
{

}

public class PressureSensor: Sensor
{

}

public class Measure
{
    public int number;
    public decimal value;
}
```

```
public interface IMeasurer
{
    decimal GetValue();
}

public class Measurer: IMeasurer
{
    private List<Sensor> sensors;
    private List<Measure> measures;

    public Measurer(List<Sensor> sens)
    {
        throw new NotImplementedException();
    }

    public decimal GetValue()
    {
        throw new NotImplementedException();
    }
}

public class UserInterface
{
    private void ShowMeasure(IMeasurer)
    {
        throw new NotImplementedException();
    }
}
```

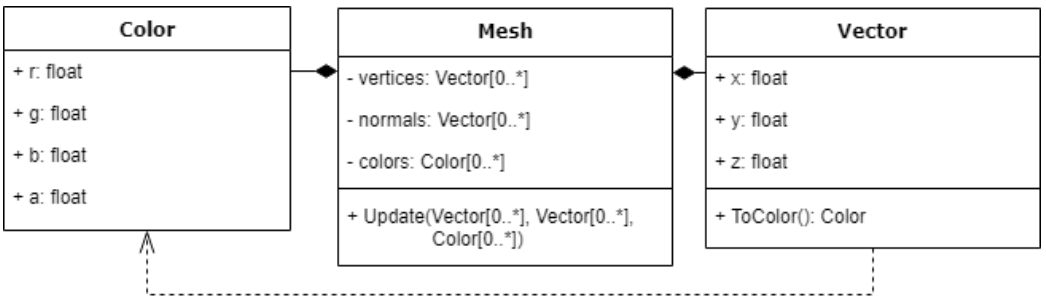
Варианты

№	Диаграммы		№	Диаграммы		№	Диаграммы		№	Диаграммы	
	1	2		1	2		1	2		1	2
1	Д8	Д14	**6**	Д9	Д12	**11**	Д5	Д11	**16**	Д3	Д4
2	Д1	Д3	**7**	Д5	Д8	**12**	Д2	Д5	**17**	Д3	Д11
3	Д2	Д14	**8**	Д2	Д7	**13**	Д3	Д12	**18**	Д6	Д13
4	Д6	Д7	**9**	Д9	Д13	**14**	Д5	Д12	**19**	Д2	Д4
5	Д9	Д14	**10**	Д4	Д8	**15**	Д6	Д8	**20**	Д12	Д13

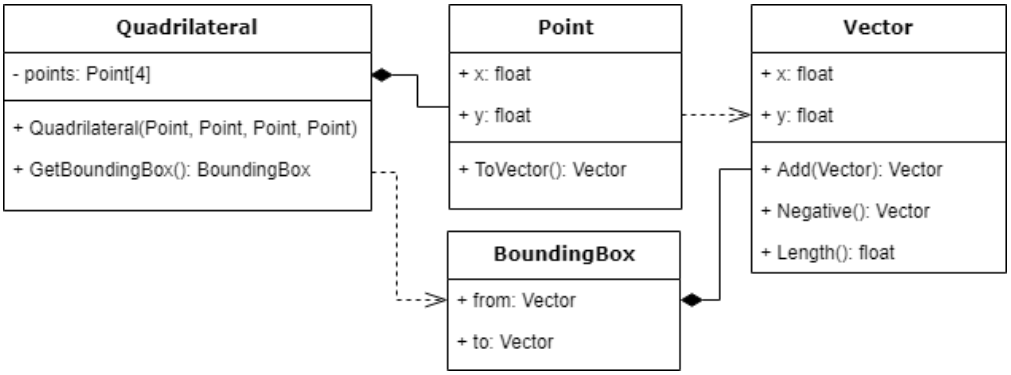
Диаграммы

№	Диаграмма
Д1	
Д2	
Д3	
Д4	

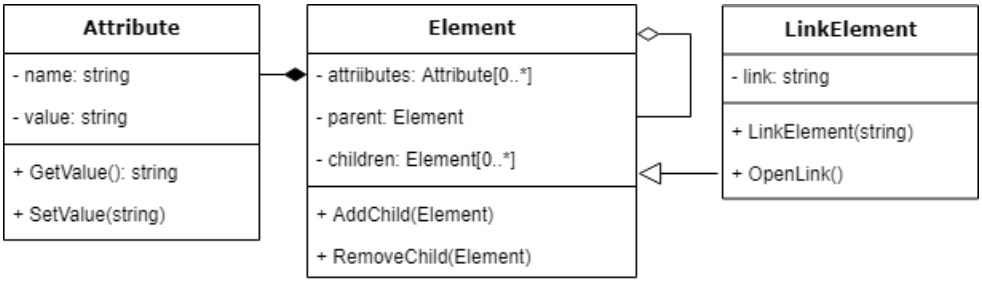
Д5


Color
+ r: float
+ g: float
+ b: float
+ a: float
Mesh
- vertices: Vector[0..*]
- normals: Vector[0..*]
- colors: Color[0..*]
+ Update(Vector[0..*], Vector[0..*], Color[0..*])
Vector
+ x: float
+ y: float
+ z: float
+ ToColor(): Color

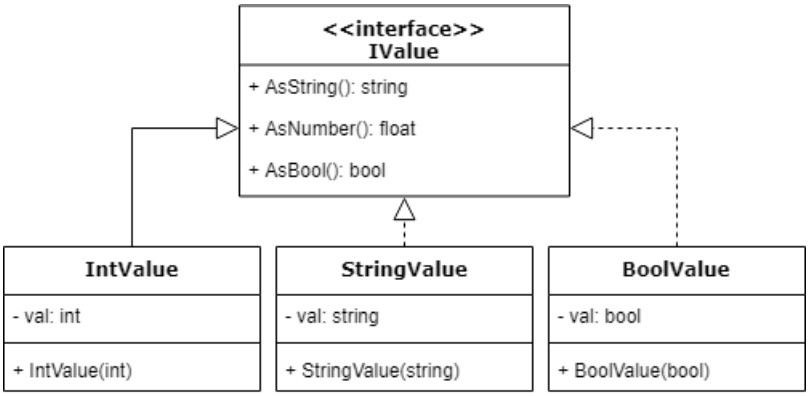
Д6


Quadrilateral
- points: Point[4]
+ Quadrilateral(Point, Point, Point, Point)
+ GetBoundingBox(): BoundingBox
Point
+ x: float
+ y: float
+ ToVector(): Vector
Vector
+ x: float
+ y: float
+ Add(Vector): Vector
+ Negative(): Vector
+ Length(): float
BoundingBox
+ from: Vector
+ to: Vector

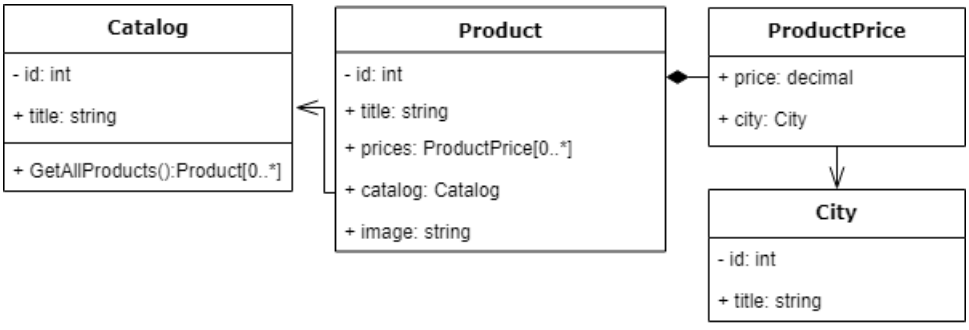
Д7


Attribute
- name: string
- value: string
+ GetValue(): string
+ SetValue(string)
Element
- attriibutes: Attribute[0..*]
- parent: Element
- children: Element[0..*]
+ AddChild(Element)
+ RemoveChild(Element)
LinkElement
- link: string
+ LinkElement(string)
+ OpenLink()

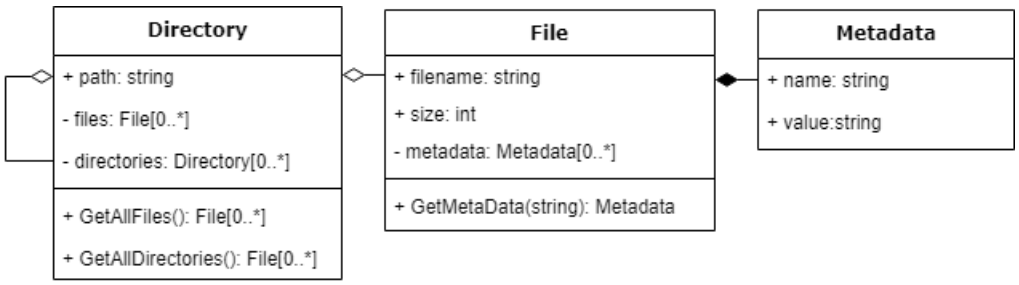
Д8


<<interface>>
IValue
+ AsString(): string
+ AsNumber(): float
+ AsBool(): bool
IntValue
- val: int
+ IntValue(int)
StringValue
- val: string
+ StringValue(string)
BoolValue
- val: bool
+ BoolValue(bool)

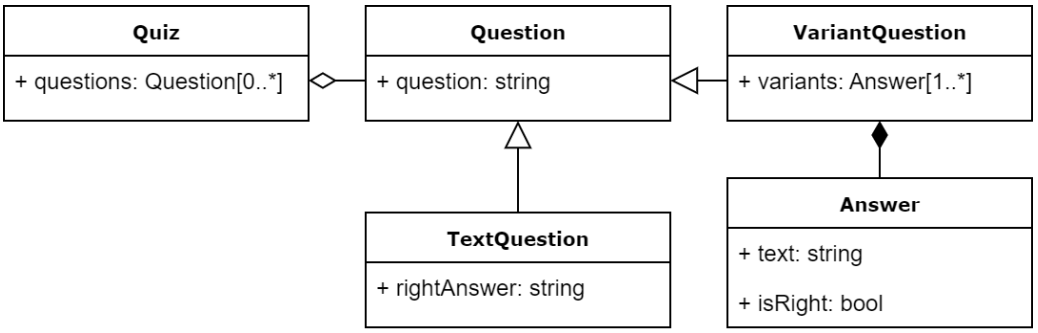
Д9


Catalog
- id: int
+ title: string
+ GetAllProducts():Product[0..*]
Product
- id: int
+ title: string
+ prices: ProductPrice[0..*]
+ catalog: Catalog
+ image: string
ProductPrice
+ price: decimal
+ city: City
City
- id: int
+ title: string

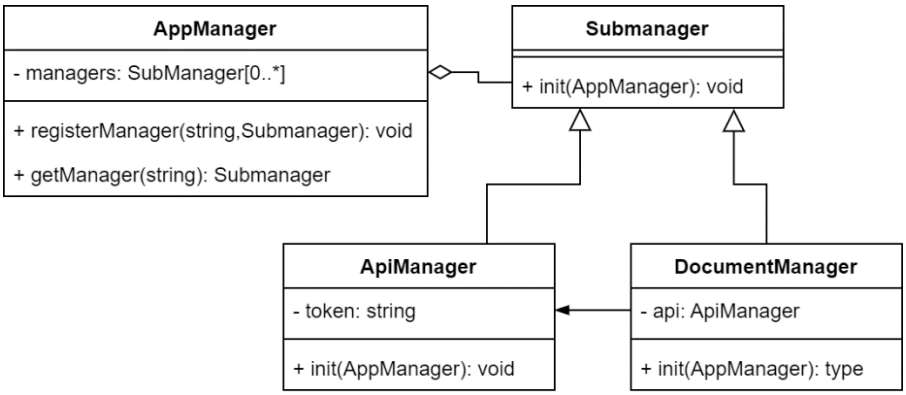
Д10



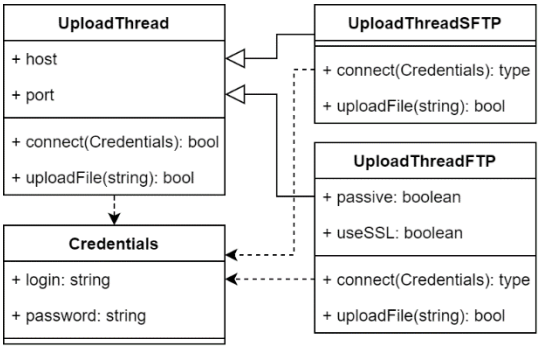
Д11



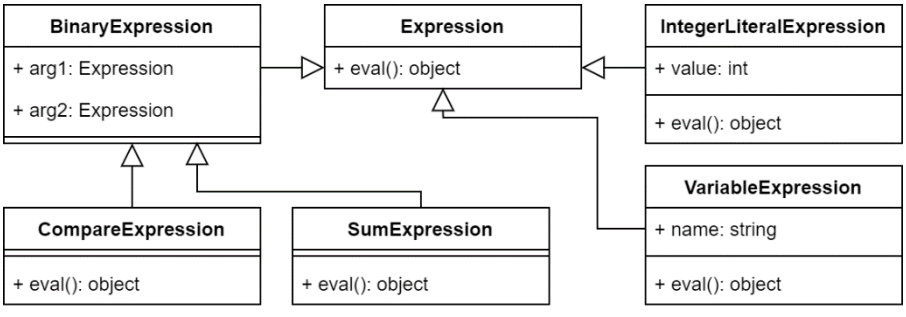
Д12



Д13



Д14



4E.2. Создать класс, реализующий заданный интерфейс

Необходимо создать консольное приложение, позволяющее вводить числа с клавиатуры и выполнять с ними заданную операцию.

В коде программы должен быть класс, реализующий следующий интерфейс:

```
interface ICalculator
{
    void AddNumber(float number);
    float Calculate();
}
```

При запуске программы должен создаваться экземпляр класса, реализующего интерфейс *ICalculator*. Далее в цикле ввода чисел каждое новое число передается в метод *AddNumber*. Как только пользователь ввел пустую строку, ввод заканчивается, и программа должна вызывать метод *Calculate*, который выполняет заданную операцию с теми числами, которые передавались в метод *AddNumber*. После расчета полученный результат должен быть выведен в консоль.

В созданном классе, реализующем *ICalculator*, не должно быть обращения к консоли. Все члены этого класса, не указанные в интерфейсе, должны иметь модификатор доступа private.

Варианты

№	Операция	№	Операция
1	У каждого второго числа поменять знак и просуммировать все числа	11	Перемножить дробные части всех чисел
2	Просуммировать дробные части всех чисел	12	Просуммировать модули всех чисел
3	Просуммировать все числа, которые меньше, чем предыдущее число	13	Умножить числа на их порядковые номера и вычислить среднее
4	Вычислить среднее из квадратов чисел	14	Просуммировать квадраты всех чисел
5	Перемножить все отрицательные числа	15	У каждого второго числа поменять знак и посчитать среднее
6	Просуммировать все числа, которые больше, чем предыдущее число	16	Просуммировать все числа, которые больше первого из чисел
7	Просуммировать все четные числа	17	Просуммировать все нечетные числа
8	Извлечь квадратный корень из суммы квадратов чисел	18	Умножить числа на их порядковые номера и сложить результаты
9	Округлить числа и сложить результаты	19	Перемножить все положительные числа
10	Разделить все числа на первое из чисел и вычислить сумму результатов	20	Вычислить среднее значение

4E.3. Разработать модульный тест для функции проверки правильности ввода данных, возвращающей код ошибки

Необходимо разработать функцию проверки правильности ввода данных для заданной записи и модульный тест, ее проверяющий. Функция проверки правильности ввода должна принимать на вход объект записи и возвращать, либо 0, если ошибок нет, либо номер поля с ошибкой (начиная с 1)

Решение должно состоять из двух проектов:

- *Библиотека классов (.NET Framework)*, в которой должен быть класс с функцией проверки правильности ввода данных и класс, представляющий собой запись
- *Проект модульного теста (.NET Framework),* в котором должен быть реализован модульный тест.

Модульный тест должен проверить, как положительный исход функции, так и все варианты отрицательного исхода.

Варианты

№	Поля записи и их проверка		
	Поле 1	**Поле 2**	**Поле 3**
1	Фамилия (строка): не меньше 3 букв	Должность (строка): только русские буквы	Оклад в руб (целое число): больше 0
2	Название цеха (строка): только буквы и цифры	План выпуска деталей (целое число): не больше 1000	Фактический выпуск деталей (целое число): не меньше 0
3	Город (строка): не больше 255 символов	Улица (строка): только русские буквы, знак дефиса, пробела и цифры	Номер дома (число): больше 0
4	Фамилия (строка): не меньше 3 букв	Рост (вещественное число): не больше 300	Вес (вещественное число): больше 30
5	Адрес отправления (строка): не меньше 30 символов	Адрес доставки (строка): не равен адресу отправления	Вес (вещественное число): не больше 90
6	Автомобильный номер (строка): только цифры и буквы А, В, Е, К, М, Н, О, Р, С, Т, У, Х	Год выпуска (целое число): от 1980 до текущего года включительно	Пробег в км (целое число): больше или равно 0
7	Название (строка): не больше 255 символов	Число сезонов (целое число): больше или равно одному	Год выпуска первого сезона (целое число): от 2000 до текущего года включительно
8	Фамилия (строка): не меньше 3 букв	Номер группы (строка): только цифры, знак дефиса и русские буквы	Номер в группе (целое число): больше 0
9	Фамилия (строка): не меньше 3 букв	Год поступления (целое число): не меньше 1952 и не больше текущего года	Средний балл (вещественное число): от 0 до 5 включительно
10	Фамилия (строка): не меньше 3 букв	Оценка за теорию (целое число): не меньше 3	Оценка за практику (целое число): не больше 5
11	Номер заказа (строка): только латинские буквы и цифры	Описание (строка): не меньше 200 символов	Сумма заказа (целое число): не меньше 0
12	Дисциплина (строка): только русские буквы, знаки дефиса и пробела	номер курса (целое число): от 0 до 6 включительно	количество часов (целое число): от 0 до 9999 включительно
13	Номер телефона (строка): только цифры, скобки, знаки тире, плюса и пробела	Имя оператора (строка): не меньше 3 символов	баланс в копейках (целое число): не меньше -9999999
14	Название товара (строка): не меньше 10 символов	Количество на складе (целое число): от 0 до 999 включительно	Количество зарезервированных (целое число): не меньше 0
15	Название материала (строка): не меньше 10 символов	Объем (вещественное число): не больше 1000	Вес (вещественное число): не меньше 0

16	Тема письма (строка): не больше 1024 символов	Адресат (строка): только латинские буквы, знак «@», точка, тире и цифры	Число слов (целое число): не меньше 1
17	Адрес сайта (строка): только латинские буквы, точка, тире и цифры	Число посетителей (целое число): не меньше числа уникальных посетителей	Число уникальных посетителей (целое число): не меньше 0
18	Производитель (строка): не меньше 10 символов	Объем выпуска (вещественное число): не меньше 0	Средняя цена (вещественное число): от 1 до 99999 включительно
19	Компания (строка), не меньше 10 символов	Сумма поступлений в млн. руб. (вещественное число): не меньше 0 и не больше 1000000	Сумма списаний в млн. руб. (вещественное число): не меньше 0
20	Фамилия (строка): не меньше 3 букв	Число ролей (целое число): не меньше 0 и не больше 9999	Гонорар в млн. руб. (вещественное число): не меньше 0

4Е.4. Прочитать метаданные из файла заданного бинарного формата

Вам необходимо разработать программу, которая считывает заданное поле метаданных (см. варианты) из указанного бинарного формата и выводит его на экран.

Обрабатываемый файл указывается пользователем при запуске программы, которая может быть, на Ваш выбор, консольной или оконной.

Возможные варианты бинарных форматов:

Формат	**Ссылки на описание**
BMP	https://ru.wikipedia.org/wiki/BMP
PNG	https://habr.com/ru/post/130472/
JPEG	https://ru.wikipedia.org/wiki/JPEG, https://habr.com/ru/post/102521/
GIF	http://home.onego.ru/~chiezo/gif.htm, https://habr.com/ru/post/274917/
WAV	http://microsin.net/programming/pc/wav-format.html, https://audiocoding.ru/articles/2008-05-22-wav-file-structure/
AVI	https://ru.wikipedia.org/wiki/Audio_Video_Interleave

Для анализа содержимого файла вы можете воспользоваться бесплатным онлайн-ресурсом https://hexed.it/

В папке https://github.com/Nordth/istu-priklad-practic-2023/tree/main/media/day4/Task_ReadBinaryFiles приведены примеры тестовых файлов, метаданные которых приведены в следующей таблице:

Файл	**Поле метаданных**	**Значение**
sun.bmp	ширина	64
	высота	58
	битность растра	24
bird.png	ширина	16
	высота	16
	битовая глубина цвета	8
	наличие альфа-канала	да
lenna.jpg	ширина	220
	высота	220
	количество каналов	3

cat.gif	ширина	64
	высота	67
	количество кадров	52
	наличие глобальной палитры	да
laugh.wav	число каналов	2
	частота дискретизации	48000
	количество бит в сэмпле	16
coding.avi	ширина	480
	высота	270
	число кадров в секунду	10

Ваша программа должна обрабатывать любые файлы заданного формата. В папке файлы приведены исключительно для примера

Ваша задача – разработать код, декодирующий файлы заданного формата. Использование готовых функций работы с изображениями/видео/аудио запрещено

Варианты

№	Формат	Поле метаданных
1	BMP	ширина
2	BMP	высота
3	BMP	битность растра
4	PNG	ширина
5	PNG	высота
6	PNG	битовая глубина цвета
7	PNG	наличие альфа-канала
8	JPEG	ширина
9	JPEG	высота
10	JPEG	количество каналов
11	GIF	ширина
12	GIF	высота
13	GIF	количество кадров
14	GIF	наличие глобальной палитры
15	WAV	число каналов
16	WAV	частота дискретизации
17	WAV	количество бит в сэмпле
18	AVI	ширина
19	AVI	высота
20	AVI	число кадров в секунду

4E.5. Разработать программу, выполняющую команды, полученные по сети

Вам необходимо разработать программу (консольную или оконную, на Ваш выбор) для обработки списка целых чисел. Команды для обработки чисел программа должна получать по сети путем создания TCP-подключения к серверу.

Программа-сервер уже написана, ее сборка и исходный код лежат по адресу https://github.com/Nordth/istu-priklad-practic-2023/tree/main/media/day4/Task_NetServer. После запуска программа-сервер программа создаст локальный сервер на порту 3005 (порт можно изменить, указав новый порт в аргументах командной строки)

В зависимости от варианта, ваша программа должна реализовывать несколько команд из следующего списка:

1. *add <number>* - добавить число <number> в список для обработки
2. *range <number1> <number2>* - добавить числа от *<number1>* до *<number2>* с шагом 1 в список для обработки
3. *rand* – добавить случайное число от -100 до 100 в список для обработки и вывести его на экран
4. *copy* – добавить в список для обработки копию последнего из ранее добавленных. Если в список числа еще не добавлялись, то добавить в список число 0
5. *clear* – очистить список чисел для обработки
6. *pop* – убрать из списка для обработки последнее из добавленных чисел и вывести его на экран
7. *mul <number>* – умножить все числа в списке для обработки на <number>
8. *neg* – поменять знак у всех чисел в списке для обработки
9. *abs* – сделать все числа в списке для обработки положительными
10. *print* – вывести на экран все числа в списке для обработки
11. *top* – вывести на экран последнее из добавленных в список для обработки чисел. Если чисел не было, то вывести «Список пуст»
12. *count* – вывести количество чисел, находящихся в списке для обработки
13. *countodd* – вывести количество нечетных чисел, находящихся в списке для обработки
14. *counteven* – вывести количество четных чисел, находящихся в списке для обработки
15. *sum* – вывести сумму чисел, находящихся в списке для обработки
16. *sumodd* – просуммировать все нечетные числа в списке для обработки
17. *sumeven* – просуммировать все четные числа в списке для обработки
18. *avg* – вывести среднее значение чисел, находящихся в списке для обработки

Для того, чтобы создать подключение к серверу выполните следующее:

```
using (TcpClient client = new TcpClient())
{
    client.Connect("127.0.0.1", 3005);
    using (NetworkStream stream = client.GetStream())
    {
        // Действия
    }
}
```

Работа с TCP-подключением похожа на работу с файлом. Общение с сервером будет происходить по бинарному протоколу, поэтому для удобства вы можете использовать классы *BinaryReader* и *BinaryWriter* с потоком *NetworkStream.* Порядок записи байт в числах – от младшего к старшему (little-endian)

После подключения вам необходимо отправить информацию, какие команды, используются в Вашем варианте. Для этого вы должны отправить 5 номеров команд в виде 5-ти однобайтовых чисел (нумерация соответствует нумерации списка выше)

После отправки перечня команд сервер отправит 4 байта – кол-во команд, который вам пришлет сервер.

После отправки числа команд сервер будет случайным образом посылать команды на исполнение. Команда на исполнение состоит из номера команды (1 байт) и, если есть, из аргументов команды (каждый аргумент занимает 4 байта)

Вам необходимо принять команду, вывести на экран ее название, аргументы и выполнить ее.

После отправки всех команд сервер разорвет соединение.

Перед завершением работы вашей программы вы должны вывести текущий список чисел

Пример

1. После подключения вы отправили номера команд: 1, 2, 7, 8, 15 (0x010207080F)
2. Сервер вам в ответ прислал, что число команд равно 4 (0x04000000)
3. Сервер отправил команду *add 6* (0x0106000000)
4. У вас в списке на обработку теперь одно чисдо – шесть
5. Сервер отправил команду *range 10 15* (0x010A0000000F000000)
6. У вас в списке на обработку теперь следующие числа: 6, 10, 11, 12, 13, 14, 15
7. Сервер отправил команду *sum* (0x0F)
8. Вы выводите сумму, равную 81
9. Сервер отправил команду *mul 2* (0x0702000000)
10. У вас в списке на обработку теперь числа: 12, 20, 22, 24, 26, 28, 30

Варианты

№	**Команды**				
1	add	rand	clear	countodd	avg
2	rand	copy	clear	print	sum
3	rand	copy	mul	print	avg
4	add	rand	neg	sumodd	avg
5	range	copy	abs	top	countodd
6	range	copy	mul	sumeven	avg
7	add	copy	abs	print	sumeven
8	rand	copy	neg	print	avg
9	range	rand	mul	top	avg
10	range	rand	clear	counteven	avg
11	add	range	pop	counteven	sumodd
12	add	copy	abs	top	sum
13	add	rand	neg	countodd	sum
14	range	copy	mul	top	count
15	add	range	clear	print	counteven
16	add	copy	neg	countodd	sumeven
17	add	rand	pop	print	count
18	add	range	pop	sumodd	avg
19	range	rand	abs	top	counteven
20	range	rand	pop	top	sumodd