I:
S: Какой элемент не является составной частью MS .NET Framework?
+: MS Office
-: Common Language Specification
-: Common Type System
-: Common Language Runtime
I:
S: К какой группе принадлежат типы int, float, double?
+: встроенные типы по значению
-: перечисления
-: самоописываемые типы
-: упакованные типы по значению
I:
S: На какой тип отображается тип float в библиотеке .NET?
+: System.Single
-: System.Double
-: System.UInt64
-: System.UInt32
I:
S: С помощью какого типа в .NET framework представляются отдельные символы?
+: System.Char
-: System.String
-: System.Byte
-: System.Decimal
-: System.SByte
I:
S: Какое пространство имен является корневым в библиотеке .NET
+: System
-: Data
-: Object
-: Input
-: IO
I:
S: В каких фрагментах кода объявляются переменные?
+: `int a;`
+: `int b = 7;`
+: `string str = "Hello, World!!!";`
-: `a = 123;`
I:
S: С помощью какого зарезервированного слова происходит объявление пространства имен?
+: namespace
-: using

-: break
-: foreach
-: name
I:
S: С помощью какого зарезервированного слова происходит подключение пространства имен для использования его типов?
+: using
-: namespace
-: break
-: foreach
-: name
I:
S: Какие элементы .NET Framework не могут являться членами типов?
+: сборка
+: пространство имен
-: вложенный тип
-: метод
-: операция
-: конструктор
I:
S: Выберите из приведенных понятий основные принципы объектно-ориентированного программирования.
+: инкапсуляция
+: наследование
+: полиморфизм
-: структурность
-: расширяемость
-: многоаспектность
I:
S: Как называется отдельный экземпляр класса?
+: объект
-: субъект
-: делегат
-: представитель
-: оболочка
I:
S: Дано определение класса на языке C#. Какие члены класса являются закрытыми?

```
public class E1
{
    int A;
    float B;
    public int C;
    private float D;
}
```

+: A
+: B
+: D
-: C
I:
S: Дано определение класса:

```
public class Point
{
    float Zz;
    private double Q;
    protected int x;
    public int y;
}
```

Какие поля класса являются закрытыми (недоступными из объектной переменной класса)?
+: Zz
+: Q
+: x
-: y
I:
S: Дано определение класса:

```
public class Человек
{
    int ПолныхЛет;
    string ФИО;

    public Человек(int pAge, string pFIO)
    {
        ПолныхЛет = pAge;
        ФИО = pFIO;
    }
}
```

и фрагмент использования класса в функции Main:

```
    static void Main(string[] args)
    {
        Человек чел = new Человек(3, "");
        чел.ПолныхЛет = 5;
        чел.ФИО = "Николаев Е.И.";
    }
```

Какие ошибки существуют в коде?
+: в функции Main осуществляется доступ к закрытым полям из объектной переменной
-: при создании объекта указаны неверные параметры
-: в определении класса отсутствует конструктор по умолчанию
I:
S: Даны определения классов на языке C#.

```
public class Parent
{
    protected int a;
}
```

```
class Child : Parent
{
    int b;

    void E()
    {
        b = a;
    }
}
```

```
class Child1 : Parent
{
    int c;

    public Child1() { }
}
```

Существуют ли ошибки в данном фрагменте?
+: ошибок нет
-: переменная a является недоступной в производном классе, следовательно, присвоение b = a некорректно
-: в двух классах отсутствуют конструкторы
-: все поля классов являются закрытыми, поэтому их использование не представляется возможным
I:
S: Что такое конструктор по умолчанию в языке C#?
+: Программист, определяя класс, может явно не реализовывать конструктор. В этом случае .NET Framework использует пустой метод в качестве конструктора по умолчанию
-: Конструктор по умолчанию представляет собой метод, который вызывается перед созданием объекта и устанавливает значения всех полей класса в предопределенные значения
-: Программист определяет сколько угодно свих конструкторов, но .NET Framework будет игнорировать их, если не отключен конструктор по умолчанию
I:
S: Даны определения классов на языке C#.

```
public class Parent
{
    protected int a;

    Parent()
    {
        a = 0;
    }
}
```

```
class Child1 : Parent
{
    int c;

    public Child1()
        :base()
    { }
}
```

Существуют ли ошибки в данном фрагменте?
+: конструктор базового класса не виден из производного класса
-: ошибок нет
-: конструктор базового класса private, а производного – public
-: все поля классов являются закрытыми, поэтому их использование не представляется возможным
I:
S: Даны определения классов на языке C#.

```
public class Parent
{
    protected int a;

    protected Parent()
    {
        a = 0;
    }
}
```

```
class Child1 : Parent
{
    int c;

    public Child1()
        :base()
    { }
}
```

Существуют ли ошибки в данных фрагментах?
+: ошибок нет
-: конструктор базового класса не виден из производного класса
-: конструктор базового класса private, а производного – public
-: все поля классов являются закрытыми, поэтому их использование не представляется возможным
I:
S: Даны определения классов на языке C#.

```
public class Parent: Child1
{
    protected int a;

    protected Parent()
    {
        a = 0;
    }
}
```

```
class Child : Parent
{
    int b;

    void E()
    {
        b = a;
    }
}
```

```
class Child1 : Child
{
    int c;
}
```

Существуют ли ошибки в данных фрагментах?
+: классы наследуются по кругу
-: неверное использование модификатора доступа protected в классе Parent
-: в двух классах отсутствуют конструкторы
-: все поля классов являются закрытыми, поэтому их использование не представляется возможным
-: ошибок нет
I:
S: Сколько наследников может быть у представленного класса?

```
public class Child : Parent
{
    int b;

    void E()
    {
        b = a;
    }
}
```

+: столько, сколько реализует программист (принципиальных ограничений нет)
-: количество наследников ограничивается в базовом классе Parent
-: не более 128
I:
S: В языке C# программист определил класс следующим образом:

```
public class Parent
{
    protected int a;

    protected Parent()
    {
        a = 0;
    }
}
```

Какой класс является базовым для данного класса?

+: System.Object

-: System.Data

-: System.IO

-: нет базового класса

I:

S: В языке C# программист определил класс следующим образом:

```
public class Parent
{
    protected int a;

    protected Parent()
    {
        a = 0;
    }
}
```

Как характеризуется переменная a по отношению ко всем производным классам?

+: в производных классах переменная a является видимой внутри класса, но недоступной из объектной переменной

-: в производных классах переменная a является недоступной как внутри класса, так и из объектной переменной

-: в производных классах переменная a является недоступной внутри класса, но доступной из объектной переменной

-: в производных классах переменная a является видимой как внутри класса, так и из объектной переменной

I:

S: В языке C# программист определил классы и создал их объекты следующим образом:

```
public class Parent
{
    protected int a;

    protected Parent(int pA)
    {
        a = pA;
    }
}
```

```
public class Child : Parent
{
    int b;

    public Child()
        :base(0)
    {
        b = a;
    }
}
```

```
static void Main(string[] args)
{
    Parent ob1 = new Parent();

    Child ob2 = new Child();
}
```

Какая ошибка была допущена?

+: нельзя создавать объект ob1, так как класс Parent не имеет открытого конструктора

-: некорректное использование зарезервированного слова base

-: в конструкторе класса Child нет доступа к переменной a базового класса
-: нельзя создать объект базового класса, так как объявлен класс-наследник
-: ошибок нет
I:
S: В языке C# программист определил классы и создал их объекты следующим образом:

```
public class Parent
{
    protected int a;

    protected Parent(int pA)
    {
        a = pA;
    }

    public Parent()
        : this(0) { }
}
```

```
public class Child : Parent
{
    int b;

    public Child()
        :base()
    {
        b = a;
    }
}
```

```
static void Main(string[] args)
{
    Parent ob1 = new Parent();

    Child ob2 = new Child();
}
```

Какая ошибка была допущена?
+: ошибок нет
-: некорректное использование зарезервированного слова base
-: в конструкторе класса Child нет доступа к переменной a базового класса
-: некорректное использование зарезервированного слова this
-: нельзя создавать объект ob1, так как класс Parent не имеет открытого конструктора
I:
S: В языке C# программист определил классы и интерфейсы следующим образом:

```
public class MainObject: IString, IDraw
{
    int data;

    public string GetStr()
    {
        return "Даннные: " + data.ToString();
    }

    public void Draw()
    {
        Console.WriteLine(data.ToString());
    }
}
```

```
public interface IDraw
{
    void Draw();
}

public interface IString
{
    string GetStr();
}
```

Какая ошибка была допущена?
+: нет ошибок
-: класс MainObject не может иметь двух базовых классов
-: некорректное использование зарезервированного слова interface
-: в классе IString метод GetStr не определен
I:

S: На языке C# программист определил класс, интерфейсы и создал объекты следующим образом:

```
public interface IDraw
{
    void Draw();
}

public interface IString
{
    string GetStr();
}

public class Parent
{
    protected int a;
}
```

```
public class MainObject: IString, IDraw
{
    int data;

    public string GetStr()
    {
        return "Даннные: " + data.ToString();
    }

    public void Draw()
    {
        Console.WriteLine(data.ToString());
    }
}
```

```
static void Main(string[] args)
{
    MainObject ob1 = new MainObject();
    ob1.Draw();

    Parent ob2 = new Parent();
    ob2.Draw();
}
```

Какая ошибка была допущена?

+: в классе Parent не определен метод Draw, который вызывается через объектную переменную ob2

-: в классе Parent нет конструктора, поэтому объект ob2 не может быть создан

-: класс MainObject не может иметь двух базовых классов

-: ошибок нет

I:

S: На языке C# программист определил класс, интерфейсы и создал объекты следующим образом:

```
public interface IDraw
{
    void Draw();
}

public interface IString
{
    string GetStr();
}
```

```
public class MainObject: IString, IDraw
{
    int data;

    public string GetStr()
    {
        return "Даннные: " + data.ToString();
    }
}
```

Какая ошибка была допущена?

+: класс MainObject не реализует метод Draw, наследуемого интерфейса IDraw

-: класс MainObject не может иметь двух базовых классов

-: класс MainObject содержит метод GetStr, который дублирует метод базового класса IString

-:ошибок нет

I:

S: На языке C# программист определил класс, интерфейсы:

```
public interface IString
{
    string GetStr();
}

public class Data1 : IString
{
    int data;

    public string GetStr()
    {
        return "Данные: " + data.ToString();
    }
}
```

```
public class Data2 : IString
{
    string data;

    public string GetStr()
    {
        return data;
    }
}
```

Затем программист реализовал функцию Main:

```
    static void Main(string[] args)
    {
        IString[] mas = new IString[] { new Data1(), new Data2() };
    }
```

Какая ошибка была допущена?

+: нет ошибок

-: в массиве присутствуют элементы различного типа

-: не указан размер массива при его создании

-: тип массива, заявленный при объявлении, и тип добавляемых элементов не совпадают

I:

S: На языке C# программист определил класс, интерфейсы:

```
public interface IString
{
    string GetStr();
}

public class Data1 : IString
{
    int data;

    public string GetStr()
    {
        return "Данные: " + data.ToString();
    }
}
```

```
public class Data2 : IString
{
    string data;

    public string GetStr()
    {
        return data;
    }
}
```

Затем программист реализовал функцию Main:

```
static void Main(string[] args)
{
    IString[] mas = new IString[] { new Data1(), new Data2() };
    IString ob = new IString();
}
```

Какая ошибка была допущена?

+: Невозможно создать объект интерфейса ob

-: нет ошибок

-: в массиве присутствуют элементы различного типа

-: тип массива, заявленный при объявлении, и тип добавляемых элементов не совпадают

I:

S: Какие из фрагментов кода C# выводит цифры от 1 до 10 в столбец?

```
for (int i = 1; i < 11; i++)
    Console.WriteLine(i.ToString());
for (int j = 1; j > 0; j++)
{
    Console.WriteLine(j.ToString());
    if (j == 10) break;
}
```
+:
```
for (int j=0; true; j++)
    Console.WriteLine(j.ToString());
```
-:
I:
S: Для чего служит оператор break?
+: прерывает выполнение текущей итерации цикла и цикла в целом
-: прерывает выполнение текущей итерации цикла и переходит к следующей итерации
-: прерывает выполнение текущей итерации цикла и переходит к выполнению всего цикла с начала
I:
S: Для чего служит оператор continue?
+: прерывает выполнение текущей итерации цикла и переходит к следующей итерации цикла
-: прерывает выполнение текущей итерации цикла и цикла в целом
-: прерывает выполнение текущей итерации цикла и переходит к выполнению всего цикла с начала
I:
S: В переменной Sum по завершении выполнения фрагмента кода будет содержаться...
```
long Sum = 0;
int i;

for (i = 1; i <= 1000000; i++)
    if (i % 3 == 0)
        Sum += i;
```
+: сумма всех чисел кратных 3 и находящихся в диапазоне от 1 до 1000000
-: сумма кубов всех чисел от 1 до 1000000
-: 0
I:
S: Какое зарезервированное слово не может присутствовать в приведенном фрагменте?
```
int a = 0;

if (a == 0)
    a = 1;
else if (a == 1)
    a = 2;
else if (a == 2)
    a = 3;
default
    a = 0;
```
+: default

-: else if
-: if
-: все зарезервированные слова могут присутствовать
I:
S: Какое зарезервированное слово не может присутствовать в приведенном фрагменте?

```
int a = 0, S = 0;

while (true)
{
    if (a == 100)
        break;
    else
        if (a % 2 == 0)
            continue;

    S += a;
    a++;
}
```

+: все зарезервированные слова могут присутствовать
-: true
-: continue
-: else
-: break
I:
S: В каком случае показана корректная инициализация размера массива в C#?
+: все фрагменты корректны
-: `mas_d = new double[5];`
-: `mas_int = new Int32[100];`
-: `MassivStr = new string[100];`
I:
S: Что такое индексатор?
+: это целочисленный номер элемента массива
-: это максимальный элемент массива
-: это размер массива
-: это размер массива, уменьшенный на 1
I:
S: Приведено объявление массива.

```
// Объявление и инициализация массива
int[] mas = new int[4];
```

Какие из приведенных фрагментов обращения к элементам массива некорректны?
+: `int Oo = mas[4];`
-: `mas[0] = 1000;`
-: `int val = mas[1];`
I:
S: Какие ошибки присутствуют в приведенном фрагменте?

```
int[] mas = new int[100];

for (int i = 0; i < 1000; i++)
    mas[i] = 1 / i;
```

+: выход за пределы размера массива
+: инициализация элементов массива значениями недопустимого типа
+: деление на 0
-: неверное объявление массива
-: цикл выполняется бесконечно

I:

S: Приведено объявление и инициализация массива.

```
int [,] mas = new int[10,10];
```

В каком фрагменте инициализируются все элементы массива?

+:
```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        mas[i, j] = i + j;
```

-:
```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        mas[i, i] = i + j;
```

-:
```
for (int i = 0; i > 10; i++)
    for (int j = 0; j > 10; j++)
        mas[i, j] = i + j;
```

-:
```
for (int i = 0; i <= 10; i++)
    for (int j = 0; j <= 10; j++)
        mas[i, j] = i + j;
```

I:

S: Объявление и инициализация массива производится следующим образом:

```
int [,] mas = new int[10,10];
int S = 0;

for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        mas[i, j] = i + j;
```

Какой фрагмент кода осуществляет вычисление суммы диагональных элементов?

+:
```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        if (i == j)
            S += mas[i, j];
        else
            continue;
```

-:
```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        if (i == j)
            continue;
        else
            S += mas[i, j];
```

-:
```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
        S += mas[i, j];
```

I:

S: Какая структура данных характеризуется поведением «первым элемент пришел – первым считывается».
+: очередь
-: стек
-: массив
-: список
I:
S: Существуют ли коллекции C# к элементам которых можно обращаться с использование индексатора?
+: Да, например List<T>
-: Да, например Queue<T>
-: Да, например Stack<T>
-: Нет, не существуют
I:
S: Чем отличается наследование реализации от наследования интерфейсов?
+: Наследование реализации предполагает, что класс-наследник реализует весь функционал базового класса, если только в базовом классе не требуется переопределить реализацию методов.
-: Наследование реализации не предполагает явной реализации поведения базового класса, а только декларирует, что класс-наследник должен реализовать методы, заявленный в базовом классе
-: Существенных отличий в типах наследования нет за исключением того, что при наследовании интерфейсов у класса потомка может быть более одного базового класса
I:
S: Приведен код на языке C#.

```
public class Parent
{
    int a;
}

class Child : Parent
{
    int b;

    void E()
    {
        b = a;
    }
}
```

Существуют ли ошибки в данном фрагменте?
+: переменная a является недоступной в производном классе, следовательно, присвоение b = a некорректно
-: в классах нет конструкторов, поэтому реализация наследования невозможна
-: переменная b является недоступной на уровне класса
-: ошибок нет
I:

S: Даны определения классов на языке C#.

```
public class Parent
{
    protected int a;
}
class Child : Parent
{
    int b;

    void E()
    {
        b = a;
    }
}
```

```
class Child1 : Parent
{
    int c;

    public Child1() { }
}
```

Существуют ли ошибки в данном фрагменте?
+: ошибок нет
-: переменная a является недоступной в производном классе, следовательно, присвоение b = a некорректно
-: в двух классах отсутствуют конструкторы
-: все поля классов являются закрытыми, поэтому их использование не представляется возможным
I:
S: Что такое конструктор по умолчанию в языке C#?
+: Программист, определяя класс, может явно не реализовывать конструктор. В этом случае .NET Framework использует пустой метод в качестве конструктора по умолчанию
-: Конструктор по умолчанию представляет собой метод, который вызывается перед созданием объекта и устанавливает значения всех полей класса в предопределенные значения
-: Программист определяет сколько угодно своих конструкторов, но .NET Framework будет игнорировать их, если не отключен конструктор по умолчанию
I:
S: Даны определения классов на языке C#.

```
public class Parent
{
    protected int a;

    Parent()
    {
        a = 0;
    }
}
```

```
class Child1 : Parent
{
    int c;

    public Child1()
        :base()
    { }
}
```

Существуют ли ошибки в данном фрагменте?
+: конструктор базового класса не виден из производного класса
-: ошибок нет
-: конструктор базового класса private, а производного – public
-: все поля классов являются закрытыми, поэтому их использование не представляется возможным
I:

S: В языке C# программист определил класс следующим образом:

```
public class Parent
{
    protected int a;

    protected Parent()
    {
        a = 0;
    }
}
```

Какой класс является базовым для данного класса?
+: System.Object
-: System.Data
-: System.IO
-: нет базового класса
I:
S: Дано определение класса на языке C#:

```
public class E1
{
    int A;
    float B;
    public int C;
    private float D;
}
```

К каким полям невозможен доступ из объектной переменной?
+: A
+: B
+: D
-: C
I:
S: Дано определение класса на языке C#:

```
public class E1
{
    int A;
    float B;
    public int C;
    private float D;
}
```

К каким полям невозможен доступ из производного класса?
+: A
+: B
+: D
-: C
I:
S: Дано определение класса:

```
public class Point
{
    float Zz;
    private double Q;
    protected int x;
    public int y;
}
```

Какие поля класса являются недоступными в производном классе?
+: Zz
+: Q
-: x
-: y
I:
S: В языке C# программист определил класс следующим образом:

```
public class Parent
{
    protected int a;

    protected Parent()
    {
        a = 0;
    }
}
```

Какие ошибки существуют в представленном коде?
+: ошибок нет
-: конструктор класса не может быть protected
-: конструктор не может обращаться к переменной помеченной как protected
-: класс не может быть помечен как public
I:
S: Дан фрагмент кода:

```
DirectoryInfo dir = new DirectoryInfo("New_Dir");
dir.Delete();
```

Что произойдет в результате его выполнения?
+: если каталог New_Dir существует и пустой, то он будет удален
-: будет удален каталог New_Dir вместе с содержимым
I:
S: Дано определение класса:

```
class Person
{
    public Person(string pFio, string pSnils, DateTime dd)
    {
        fio = pFio; snils = pSnils; birth = dd;
    }

    public override string ToString()
    {
        return "This is person \n" +
            $"FIO: {fio}, SNILS: {snils}, date of birth
              {birth.ToString()}\n";
    }

    protected string fio;
    protected string snils;
    protected DateTime birth;
}
```

Какие ошибки (если есть), допущены в представленном коде?
+: ошибок нет
-: нет конструктора по умолчанию
-: метод с зарезервированным именем ToString() не может быть определен внутри класса
-: у класса Person нет базового, а в коде предпринята попытка переопределить метод ToString()