Лабораторная работа №1

«Типы данных, классы»

**Цели работы:**

1. Научиться работать с переменными разных типов данных CTS средствами языка C#.
2. Научиться создавать классы и поля классов, инициализировать свойства классов. Научиться создавать перегруженные конструкторы классов.
3. Научиться создавать тесты для реализованных методов и классов.

**Задание №1**

Выведите на консоль минимальные и максимальные значения для предопределенных типов данных CTS.

**Задание №2**

Создайте класс с именем Rectangle.

В теле класса создайте два поля, описывающие длины сторон double sideA, sideB.

Создайте пользовательский конструктор Rectangle(double sideA, double sideB), в теле которого поля sideA и sideB инициализируются значениями аргументов.

Создайте два private метода, вычисляющие площадь прямоугольника - double CalculateArea() и периметр прямоугольника - double CalculatePerimeter ().

Создайте два свойства double Area и double Perimeter с одним методом доступа get, вызывающим созданные ранее методы.

Напишите программу, которая принимает от пользователя длины двух сторон прямоугольника и выводит на экран периметр и площадь. Покройте тестами методы класса Rectangle.

**Задание №3**

Создайте классы Point и Figure.

Класс Point должен содержать два целочисленных поля с координатами точки.

Создайте два свойства с одним методом доступа get.

Создайте пользовательский конструктор, в теле которого проинициализируйте поля значениями аргументов.

Класс Figure должен содержать конструкторы, которые принимают от 3-х до 5-ти аргументов типа Point, а также строковое автосвойство для хранения названия фигуры. Используйте ключевое слово this для вызова перегруженных конструкторов, избегайте дублирования кода.

Создайте два метода: double LengthSide(Point A, Point B), который рассчитывает длину стороны многоугольника; double PerimeterCalculator(), который рассчитывает периметр многоугольника.

Напишите программу, которая выводит на экран название и периметр многоугольника. Покройте тестами методы класса Figure.

**Теоретические сведения**

**Члены класса**

Класс содержит члены, виды членов класса описаны в следующей таблице:

| **Член класса** | **Описание** |
| --- | --- |
| Поле | Поле является элементом данных класса. Это переменная типа, который является членом класса. |
| Константа | Константы связаны с классом (хотя у них нет статического модификатора). Компилятор заменяет константы везде, где они используются, реальными значениями. |
| Метод | Методы — это функции, связанные с определенным классом. |
| Свойство | Свойства — это наборы функций, к которым клиент может получить доступ аналогично общедоступным полям класса. C# предоставляет специальный синтаксис для реализации свойств чтения и записи в ваших классах, поэтому вам не нужно использовать имена методов с префиксом Get или Set. |
| Конструктор | Конструкторы — это специальные функции, которые вызываются автоматически при создании экземпляра объекта. Они должны иметь то же имя, что и класс, к которому они принадлежат, и не могут иметь тип возвращаемого значения. Конструкторы полезны для инициализации. |
| Индексатор | Индексаторы позволяют обращаться к вашему объекту так же, как к массивам. |
| Оператор | Операторы, в самом простом виде, представляют собой такие действия, как + или –. Когда вы складываете два целых числа, вы, строго говоря, используете оператор + для целых чисел. C# поддерживает перегрузку операторов |
| Событие | События — это члены класса, которые позволяют объекту уведомлять подписчика всякий раз, когда происходит что-то примечательное, например, изменение поля или свойства класса или какая-либо форма взаимодействия с пользователем. У клиента может быть код, известный как обработчик событий, который реагирует на событие. |
| Деструктор | Синтаксис деструкторов или финализаторов подобен синтаксису конструкторов, но они вызываются, когда среда CLR обнаруживает, что объект больше не нужен. Они имеют то же имя, что и класс, с предшествующей тильдой (~). Невозможно точно предсказать, когда будет вызван финализатор. |
| Тип | Классы могут содержать внутренние классы. |

Пример класса Person с полями, свойствами, автосвойством, перегруженными конструкторами и методом:

