Семинар 2.

TODO задачи.

*Все классы реализовывайте в отдельной библиотеке классов MyLib*

№1. Реализовать класс **Robot**, содержащий функционал для управления перемещениями робота.

Закрытые поля *int* **x, y** – хранят текущее положение;

Методы **Right()**, **Left()**, **Forward()** и **Backward()** изменяют соответствующую координату в направлении движения (**right** увеличивает **x** на единицу, **backward** уменьшает y на единицу), метод **Position()** возвращает строку с текущим положением робота.

В основной программе получать программу для робота в виде строки **S**. Каждая команда кодируется заглавной латинской буквой: **R (Right), L (Left), F (Forward), B (Backward)**. В многоадресный делегат сохранять методы, в порядке, определённом программой **S**.

Запускать программу и выводить исходные и конечные координаты.

**ToDO**: Разработайте для робота консольный интерфейс. Клетки – позиции текстового курсора на экране.

• Ограничения координат на поле получать от пользователя перед запуском робота.

• Программу в виде строки S получать от пользователя (см. предыдущий слайд).

• Робот отображается символом ‘**\***’ красного цвета.

• Позиции, в которых побывал робот отмечаются символом ‘**+**’ серого цвета.

• Если программа робота выводит его за пределы поля – останавливать выполнение программы и сообщать об этом.

№2. Реализовать класс **Car** для представления автомобиля.HG

Класс содержит:

Автореализуемые свойства *int* **CurrentSpeed** – текущая скорость автомобиля, *int* **MaxSpeed** – максимальная скорость, *string* **Name** – имя автомобиля;

Закрытое поле *bool* **carIsDead** – работоспособна ли машина;

Максимальная скорость по умолчанию (пустой конструктор) – 100, конструктор с параметрами (задают скорость текущую и максимальную, имя);

ToDO:

1. Определите тип-делегат, который будет использоваться для отправки оповещений в вызывающий код

*public* *delegate* *void* **CarEngineHandler** (*string* **msgForCaller**)

1. Добавим в класс **Car** закрытое поле *private* **CarEngineHandler** **listOfHandlers**
2. Описать вспомогательную функцию в классе **Car**, позволяющую передавать метод, который должен запускаться в вызывающем коде

*public* *void* **RegisterWithCarEngine** (**CarEngineHandler** **methodToCall**)

1. реализовать метод **Accelerate()**, в котором происходят вызовы методов из вызывающего кода через делегат **listOfHandlers**.

№3. Создадим библиотеку классов с именем **Numerical**.

В библиотеке опишем метод поиска вещественного корня функции одного аргумента на заданном интервале.

Используем в качестве прототипа для решения нашей задачи алгоритм под номером 4б «Нахождение корней непрерывной функции методом деления интервала пополам» из книги «Библиотека алгоритмов 1б-50б (Справочное пособие.)» М., «Сов. радио», 1975. -176 с.

Метод *бисекции* (https://ru.wikipedia.org/wiki/Метод\_бисекции)

ToDO:

Создадим консольное приложения для тестирования библиотеки классов **Numerical**.

Найдем корни математических функций, используя библиотечный метод **bisec()**.

В качестве аргументов, заменяющих параметр-делегат, используем:

• библиотечную функцию (метод из стандартной библиотеки),

• статический метод, явно определенный в программе,

• анонимный метод,

• лямбда-выражение.

\* (задание со звёздочкой)

Дополним библиотеку **Numerical** численным методом для поиска минимума одномерной вещественной функции.

Выберем алгоритм на основе метода золотого сечения, описанный в работе

NUMERICAL METHODS for Mathematics, Science and Engineering, 2nd Ed, 1992 Prentice Hall, Englewood Cliffs, New Jersey, 07632, U.S.A. Prentice Hall, Inc.; USA, Canada, Mexico ISBN 0-13-624990-6 Prentice Hall, International Editions: ISBN 0-13-625047-5 Существует реализация алгоритма на языке Си: NUMERICAL METHODS: C Programs, (c) John H. Mathews 1995.

ToDO:

Для тестирования метода **Optimum\_1** создадим консольное приложение и запрограммируем поиск минимума следующих функций (используйте лямбда выражения):