**Задание 1.**

Написать игру со следующим сценарием.

В игре 2 участника:

- компьютер

- игрок

У игрока есть некоторая сумма денег - задается случайным образом в диапазоне от 100 до 500 рублей в начале игры. Компьютер загадывает число в диапазоне от 1 до 100. Игрок сообщает, за сколько попыток (N) он берется отгадать это число.

Если игрок угадал число за N или менее попыток, сумма на его счету увеличивается. Увеличение суммы (P) должно быть тем больше, чем меньше число N.

Если игрок не угадал число за N попыток, сумма на его счету уменьшается. Уменьшение суммы (M) должно быть тем больше, чем больше число N.

Игра завершается, когда игрок выиграет 1000 рублей или когда он все проиграет.

Подумайте, какие зависимости P и М от N надо задать, чтобы в выигрыше в основном был игровой клуб (компьютер).

**Для генерирования случайных чисел используйте класс Random.**

**Задание 2.**

Разработать класс, описывающий некоторый физический материал.

**Поля**: имя, плотность.

**Методы**: getters/setters;

ToString( ) – значения полей объекта в csv формат (например, steel;7850).

Разработать класс, описывающий *однородный* предмет, состоящий из *одного* материала.

**Поля**: имя, материал (объект класса, описанного выше), объём.

**Методы**:

* getters/setters;
* GetMass( ) – вычисление массы предмета (= плотность \* объём);

ToString( ) – объект в строку в csv формате (wire;steel;7850;0.03;235.5).

В методе Main:

1 Создать объект Стальной\_Провод из стали в объёме 0.03м3.

2 Вывести объект на консоль с использованием ToString( ).

3 Изменить материал провода на медь (плотность = 8500) и вывести на консоль его новую массу.

**Задание 3.**

Создать класс «**Окружность**», описывающий объекты – окружности на координатной плоскости. Класс должен содержать указанные ниже элементы.

* Закрытые поля для хранения координат центра окружности и радиуса.
* Конструктор без параметров для создания окружности с центром в начале координат и единичным радиусом.
* Конструктор с параметрами для создания произвольной окружности. Предусмотреть проверку на корректность введенных данных.
* Свойства для доступа к полям класса (только для чтения).
* Свойства для определения длины окружности.
* Метод, результатом которого является **true**, если окружность целиком лежит в одной координатной четверти, и **false** в противном случае.
* Метод для перемещения окружности по вертикали вниз или по горизонтали влево (в зависимости от значения соответствующего параметра) на заданную величину.
* Метод для увеличения радиуса окружности на заданную величину.
* Статический метод для проверки, пересекаются ли две окружности (входные параметры – объекты класса, результат **true** или **false).**

Разработать программу, выполняющую следующие действия:

* + Создает **три** объекта класса «**Окружность**» (один с помощью конструктора без параметров и два произвольных);
  + Выводит информацию об окружностях в виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Центр | Радиус | Длина | Лежит ли в одной координатной плоскости |
| 1 | (0;0) | 1 | 6.28 | нет |
| 2 | (3;5) | 2 | 12.56 | да |

* + Определяет, пересекаются ли какие-нибудь из данных окружностей;
  + Осуществляет перемещение или увеличение (по выбору пользователя) для первой окружности и выводит новую информацию о ней.

**Задание 4.**

Создать класс «**Функция**», описывающий объекты - функции из семейства функций заданного вида . Класс должен содержать следующие элементы:

* Закрытые поля для хранения значения параметров функции a и b.
* Свойства для доступа к параметрам функции.
* Метод для вычисления значения функции (входной параметр – значение аргумента х).
* Метод для табулирования функции (входные параметры - хнач, хкон, шаг ∆х). При хнач <= хкон шаг прибавлять, при хнач > хкон шаг вычитать (использовать тернарную операцию **? :**).
* Метод **ToString()**, результатом которого является строковое представление функции.
* Статический метод для ввода начального значения аргумента, конечного значения аргумента и шага изменения аргумента функции (выходные параметры - хнач, хкон, шаг ∆х) Ввод продолжать до тех пор пока не будет введен шаг **∆х>= 0**.

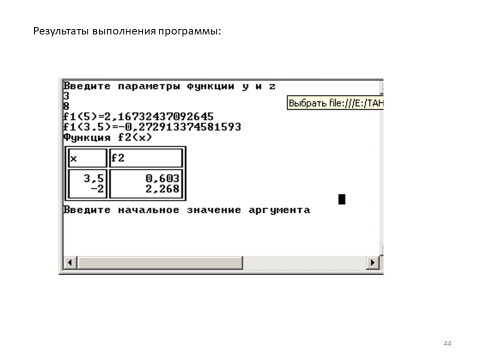
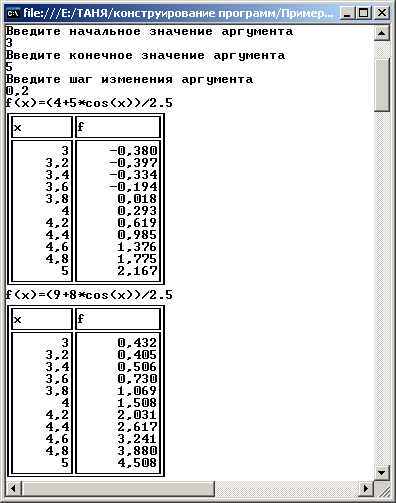
Создать класс «**Таблица**», описывающий объекты-таблицы из двух столбцов, содержащий следующие элементы:

* Закрытые поля: заголовок таблицы, заголовки столбцов, ширина первого столбца, ширина второго столбца.
* Конструктор с параметрами.
* Метод для вывода шапки таблицы.
* Метод для вывода строки таблицы (входные параметры – числовые значения, которые выводятся в строке таблицы).
* Метод для вывода низа таблицы.

Разработать программу, которая выполняет следующие действия:

* + Создает два объекта класса «**Функция**» (параметры функций вводятся с клавиатуры).
  + Для каждого объекта вычисляет значения для трех различных значений аргумента.
  + Для каждого объекта–функции выполняет табулирование для ряда значений аргумента. Результат выводить в виде таблицы, в заголовке таблицы приводить вид функции.

Ниже приведен пример работы программы.



Вопросы для самоконтроля:

**Введение в платформу Microsoft .NET**

* + Архитектура платформы Microsoft .NET.
  + Общеязыковая среда исполнения CLR (common language runtime).
  + Стандартная система типов CTS (common type system).
  + Стандартная языковая спецификация CLS (common language specification).
  + Библиотека классов FCL (BCL).
  + Языки платформы Microsoft .NET.
  + Схема компиляции и исполнения приложения платформы Microsoft .NET.
  + Язык MSIL (Microsoft Intermediate Language).
  + Понятия метаданных, манифеста, сборки.
  + Что такое рефлектор?
  + Необходимость использования рефлектора.
  + Обзор существующих рефлекторов.
  + Что такое дотфускатор?
  + Необходимость использования дотфускаторов.

**Базовые конструкции языка программирования С#**

* Анонимные типы
* Ввод, вывод в консольном приложении.
* Структурные(значимые) и ссылочные типы.
* Упаковка, распаковка
* Nullable- типы
* Цикл foreach.

**Понятие класса. Основные элементы класса**

1. Синтаксис объявления класса.
2. Спецификаторы доступа языка программирования C#.
3. Поля класса.
4. Методы класса. Виды параметров
5. Конструкторы
   1. Понятие конструктора.
   2. Параметризованный конструктор.
   3. Перегруженные конструкторы.
   4. Статические конструкторы.
6. Свойства
   1. Что такое свойства?
   2. Синтаксис объявления свойств.
   3. Примеры использования свойств.
   4. Автоматические свойства (auto-property) (Это на занятиях не успели рассмотреть ☹ Пока познакомьтесь самостоятельно)
      * Что такое автоматические свойства?
      * Примеры использования автоматических свойств

Инициализация автоматических свойств