**Группа** ИС 3-1

**ФИО:** Мозыркий А.Д.

**Курс:** C# Professional

**Урок 2:** Системные коллекции

**Теория**

* Строка (String) является упорядоченной коллекцией символов Юникода, используемой для представления текста. Объект String является упорядоченной коллекцией объектов System.Char, представляющей строку. Значением объекта String является содержимое упорядоченной коллекции, и это значение является неизменяемым (т. е. доступным только для чтения).
* Новый экземпляр объекта String можно создать следующими способами:
  + Путем присвоения строкового литерала переменной String.
  + Путем вызова конструктора класса String.
  + С помощью оператора сцепления строк (+) для создания одной строки из любой комбинации экземпляров String и строковых литералов.
  + Путем извлечения свойства или вызова метода, который возвращает строку.
  + Путем вызова метода форматирования для преобразования значения или объекта в строковое представление.
* StringBuilder - Данный класс предоставляет подобный строке объект, значение которого является изменяемой последовательностью знаков. Значение считается изменяемым потому, что после создания его можно изменить путем добавления, удаления, замены или вставки знаков.
* Емкостью StringBuilder считается максимальное количество знаков, которое экземпляр может хранить в любой момент времени. Емкость больше или равна длине строкового представления значения экземпляра.
* Регулярные выражения — это незаменимый инструмент для многих приложений, в которых ведется работа со строками или анализ объемных блоков текста. Сравнивая с текстом регулярные выражения, состоящие из чисел, букв в определенном регистре или шестнадцатеричных строк, можно принимать решения, влияющие на работу программы.
* Основа обработки текста с помощью регулярных выражений — это подсистема обработки регулярных выражений, представленная в платформе .NET Framework объектом System.Text.RegularExpressions.Regex. Минимальный набор сведений, который требуется предоставить подсистеме обработки регулярных выражений для обработки текста, сводится к: созданию шаблона регулярного выражения и применение его к анализируемому тексту.
* Основные метасимволы, для составления шаблона поиска:
  + \d - Определяет символы цифр.
  + \D - Определяет любой символ, который не является цифрой.
  + \w - Определяет любой символ цифры, буквы или подчеркивания.
  + \W - Определяет любой символ, который не является цифрой, буквой или подчеркиванием.
  + \s - Определяет любой непечатный символ, включая пробел.
  + \S - Определяет любой символ, кроме символов табуляции, новой строки и возврата каретки.
  + . - Определяет любой символ кроме символа новой строки.
  + \. - Определяет символ точки.
* Методы класса Regex, позволяют определить, встречается ли во входном тексте шаблон регулярного выражения. Для этого можно использовать метод IsMatch. Также можно извлечь из текста одно или все вхождения, соответствующие шаблону регулярного выражения, путем вызова метода Match или Matches. Первый метод возвращает объект Match, предоставляющий сведения о совпадении в тексте. Второй метод возвращает коллекцию MatchCollection, в которую входят объекты Match для всех совпадений, найденных в проанализированном тексте.
* Заменить текст, соответствующий шаблону регулярного выражения, можно путем вызова метода Replace.
* IsMatch - метод возвращающий bool. True - в случае, если шаблон соответствует строке или false - в противном случае. Метод IsMatch - сравнивает принимаемую в первом параметре строку с шаблоном.
* При проверке вводимых данных рекомендуется начинать регулярные выражения с символа «^» и заканчивать их символом «$». Это гарантирует проверку строки, точно соответствующей заданному шаблону, а не просто содержащей его.
* Кодирование — это процесс преобразования набора символов Юникода в последовательность байтов. Декодирование, наоборот, представляет собой процесс преобразования последовательности закодированных байтов в набор символов Юникода.
* Все данные в текстовых строках и файлах кодируются с использованием одного из стандартов видов кодирования. Почти всегда .NET Framework обрабатывает данные в разных кодировках автоматически. Кодированием-декодированием приходится управлять вручную, в случаях, когда выполняется взаимодействии с унаследованными или UNIX-системами, выполняется чтение-запись файлов на других языках, создаются HTML-страницы, генерируются сообщения электронной почты.
* Класс System.Text.Encoding поддерживает статические методы для кодирования декодирования текста.
* Метод System.Text.Encoding.GetEncoding возвращает объект, представляющий текст в заданной кодировке. Метод Encoding.GetBytes преобразует строку Unicode в серию байтов с заданной кодировкой.
* Чтобы задать кодировку при записи/чтении файла используют перегруженный конструктор Stream, принимающий объект Encoding.

**Практика**

**Задание 1**

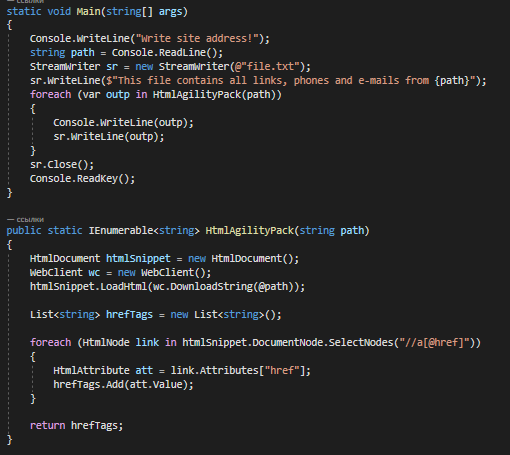


Рис.1.1 Реализация кода программы

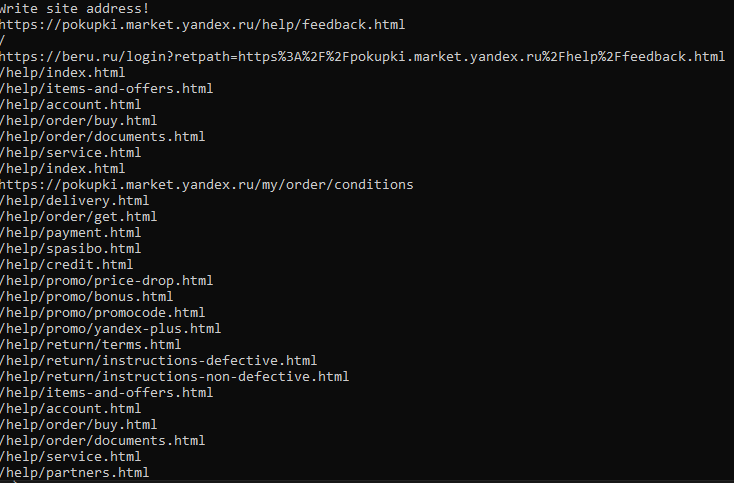




Рис.1.2 Результат работы программы

Результат работы программы соответствует ожидаемому.

**Задание 2**

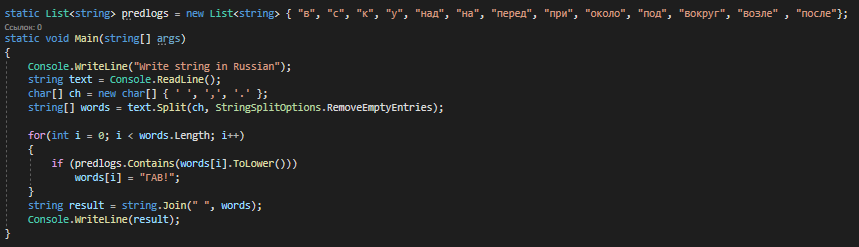


Рис.2.1 Реализация кода программы



Рис.2.2 Результат работы программы

Результат работы программы соответствует ожидаемому.

**Задание 3**

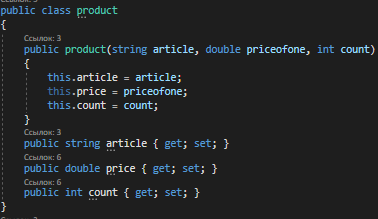


Рис.3.1 Реализация класса product

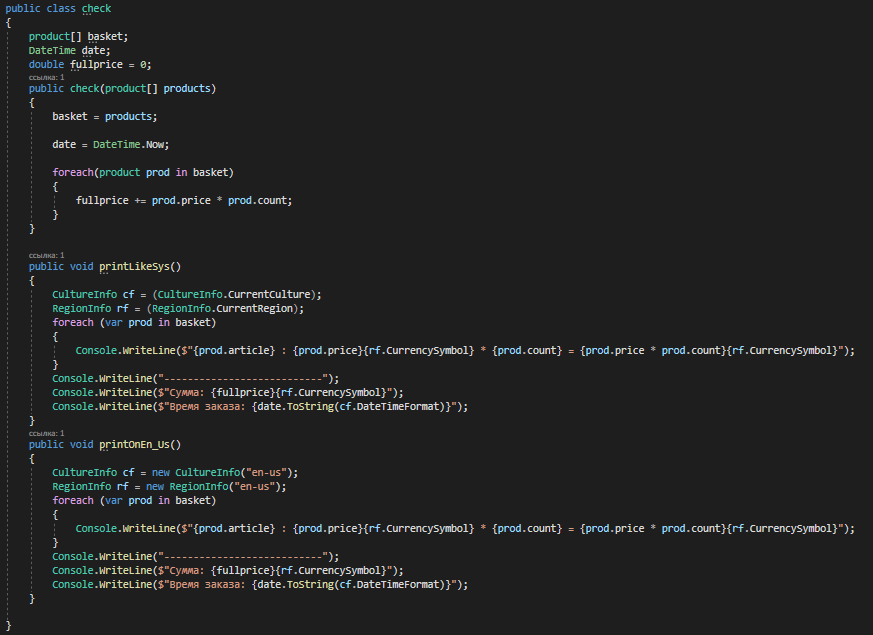


Рис.3.2 Реализация класса check

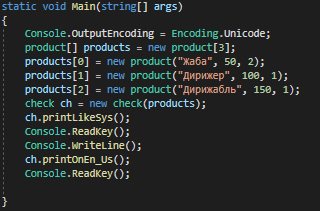


Рис.3.3 Реализация кода программы

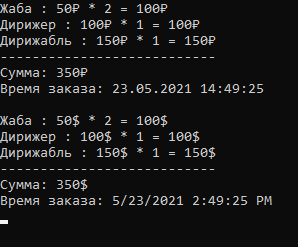


Рис.3.4 Результат работы программы

Результат работы программы соответствует ожидаемому.