**Задание 12.**

В интегрированной среде разработки *Microsoft* *Visual Studio* на языке *Visual C#* в консольном режиме составить программное обеспечение для решения типовых задач программирования по тематике «Файлы» («*Files*»), заданных по варианту.

Продумать организацию программного обеспечения, в котором чтение файла и его запись логически разделены пользовательским выбором пункта текстового меню. Предусмотреть выход из приложения.

**Элементы рабочего задания:**

1. **Учесть все возможные ограничения**, накладываемые не только на переменные, но и на участвующие в расчёте функциональные зависимости**[[1]](#footnote-1)**.

2. **Ввести контроль исходных данных**. Реализовать схему «ввод до победного» без возможности прерывания. При некорректном вводе исходных данных заставлять пользователя вводить сведения до тех пор, пока он не введёт их корректно. Продумать побуждающие сообщения-подсказки, направляющие пользователя ко вводу корректных значений.

Использовать для контроля исходных данных различные состояния метода «*TryParse*», входящего в перечень доступных компонентов интересующего значащего (*valuable*) типа данных, например: *int, byte, float, double*.

3. **Подобрать корректные тестовые примеры в необходимом количестве[[2]](#footnote-2).** Тестовые примеры хранятся в файле / в файлах. Содержимое файлов в обязательном порядке должно быть продемонстрировано в отчёте в форме скриншотов.

4. **Помнить о точности производимых вычислений**, например:

|  |  |
| --- | --- |
| **«При заданной точности  , но ,**  **то есть , а ».** | (1) |

5. Код составленного программного обеспечения **сопроводить комментариями** (элементы программистской этики)**[[3]](#footnote-3)**.

6. **Отчёт по решённой задаче составить** по схеме, изложенной в методических указаниях**[[4]](#footnote-4)**. Для удобства работы обучающихся далее представлена памятка основных разделов отчёта:

0. Титульный лист

1. Цель работы

2. Формулировка задачи

3. Блок-схема алгоритма

4. Подбор тестовых примеров

5. Листинг (код) программы

6. Расчёт тестовых примеров на ПК

7. Вывод по работе

7. **Схему алгоритма оформлять** согласно правилам**[[5]](#footnote-5)**. Помнить, что при работе с методами необходимо вводить нумерацию блоков и рассматривать содержимое методов в отдельных схемах алгоритма. Помнить, что оконечные блоки методов помимо надписей «Начало» и «Конец» должны содержать сигнатуры методов: либо краткие, в которых присутствует только наименование метода без перечисления формальных параметров (все параметры перечисляются далее в блоке «Данные» внутри метода); либо полные, в которых перечисляются все формальные параметры и их типы данных (в этом случае блок «Данные» внутри метода не размещается и не заполняется). Для ввода и вывода элементов одномерных массивов может быть применена упрощённая схема иллюстрирования: типовой фрагмент «циклический ввод» – заменён одним блоком «Данные», типовой фрагмент «циклический вывод» – заменён одним блоком «Документ».

8. Материалы, подготавливаемые **к отправке по электронной почте**, оформляются согласно **Приложению 4** Сборника задач**[[6]](#footnote-6)**. Если материалы направляются на проверку через платформу *GitHub*, то **Приложением 4** разрешается не руководствоваться.

9. Настоятельно рекомендовано перед решением задачи **ознакомиться с рекомендациями**, собранными в **Приложении 3** Сборника задач**[[7]](#footnote-7)**.

10. Не забывать **давать значимые имена переменным**, проектам, решениям, файлам, методам и другим разрабатываемым компонентам**[[8]](#footnote-8)**. При нежелании использовать значимые имена переменных в отчёте необходимо составить в разделе, содержащем формулировку задачи, таблицу соответствия вида:

Таблица 1 – Пример таблицы соответствия переменных, используемых в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Наименование переменной в программе** | **Смысловое содержание используемой переменной** | **Тип данных** |
| 1 | *a* | Ускорение | Вещественное |
| 2 | *v* | Скорость | Вещественное |
| 3 | *S* | Пройдённый путь | Вещественное |
| 4 | *t* | Затраченное время | Вещественное |
| 5 | *x* | Абсцисса точки на Декартовой плоскости | Целое |
| 6 | *y* | Ордината точки на Декартовой плоскости | Целое |
| 7 | *z* | Аппликата точки в пространстве | Целое |

11. **Учесть особенности работы с файлами.** Не забыть подключить модули: «**System.IO;**» для работы с файлами и «**System.Text;**» для работы с кодировкой.

12. **Ввести контроль существования файла при чтении**. Если файла по каким-либо причинам не существует «**File.Exists(…);**» – выдать сообщение об ошибке и не переходить к алгоритмам чтения. **В качестве пути к файлу по умолчанию принять каталог запуска приложения: Environment.CurrentDirectory.**

13. **Отключить дописывание файлов (значение признака «*append*» должно быть сброшено – «*false*»), если постановкой задачи не предусмотрено иное.**

14. **Следить за кодировкой в текстовом файле. Она должна определяться как «*ANSI*» (*Windows-1251*) и не должна определяться как «*UTF-8*» (*Unicode*). «Encoding.GetEncoding(1251)». В отдельных случаях может потребоваться дополнительная регистрация поставщика кодировки: «Encoding.RegisterProvider(CodePagesEncodingProvider. Instance);».**

**Внимание! При выявлении противоречивых условий среди вариантов индивидуального задания не предпринимать самостоятельных решений по устранению противоречий, а обратиться за консультацией к преподавателю, читающему курс «Программирования и основ алгоритмизации».**

**Задание состоит только из Индивидуальной части. Общая часть в нём отсутствует.**

**Вариант №1**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу, которая:

– считывает предложения из заранее подготовленного текстового файла (имя файла с расширением «***txt***» должно обязательно завершаться конструкцией «**\_01**») с заранее подготовленным содержимым,

– выводит считанные из файла предложения в обратном порядке в консоль.

Содержимое файла должно быть достаточным для организации постраничного вывода текста в консоль, при котором количество строк текста заведомо превышает количество доступных строк в окне консоли без необходимости перемещения прокруткой (скроллом). По заполнении доступных строк окна консоли текстом приостанавливать вывод до нажатия пользователем любой клавиши на клавиатуре. Выводить побуждающее сообщение вида: «Вывести текст далее…». При подтверждении пользователем перехода к следующей странице текста очищать содержимое консоли.

**Вариант №2**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* создать консольную программу для учебных целей. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*).

Создать в каталоге решения (*solution*) «*Debug*», вложенном в каталог «*bin*», текстовый файл с именем «***nmbrs.txt»*** и записать в него вручную (не программно) простые числа из диапазона от 1 до 500 через запятую «,» (таблица простых чисел размещена в **Приложении**). Суть ручного заполнения заключается в том, чтобы не составлять дополнительный алгоритм расчёта простых чисел при выполнении данного задания. Составление программного обеспечения для расчёта простых чисел допускается, но оно не должно становиться частью подготавливаемого отчёта по данной работе. Считать содержимое файла «***nmbrs.txt***» и вывести в консоль каждое третье значение, записанное через точку с запятой «;». По завершении чтения файла через пустую строку после выведенной конструкции вывести сумму значений, которые не приняли участия в выводе в консоль.

**Вариант №3**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). В учебной программе:

– дан массив строк, представляющий собой константу и содержащий наименования всех цветов радуги: «красный», «оранжевый», «жёлтый», «зелёный», «голубой», «синий», «фиолетовый»;

– выполняется запись в текстовый файл с именем «***clrs.txt***», создаваемый в каталоге решения (*solution*) «*Debug*», вложенном в каталог «*bin*», всех элементов заданного массива построчно (каждый элемент в новой строке) в случайном порядке без повторений;

– по завершении компоновки файла «***clrs.txt***» считывается его содержимое и выводится в консоль.

**Вариант №4**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Для нормальной работы учебной программы взять любой текстовый файл с произвольным именем (создать файл вручную / скачать файл в Интернет / найти существующий файл на персональном компьютере), разместить его в каталоге проекта «*Debug*», вложенном в каталог «*bin*». Обязательно указать в разделе отчёта с тестовыми примерами способ получения текстового файла. При помощи учебной программы:

– найти в текстовом файле величину самого длинного предложения;

– сохранить в строковую переменную с именем «*theMost*» только это предложение;

– вывести самое длинное предложение в консоль;

– вывести количество знаков препинания в самом длинном предложении (с учётом того, которым завершается предложение);

– вывести количество гласных букв в составе самого длинного предложения.

**Вариант №5**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл с именем «***storage01.txt***» по пути «*../bin/Debug/*» решения (*solution*). В каждой строке текстового файла последовательно записаны: код, название, количество, стоимость товаров некоторой фирмы. Все значения для одного товара разделены между собой символом доллар «$». Каждый товар в файле занимает строку (сколько товаров – столько и строк). В результате работы учебной программы:

– в текстовый файл с именем «***storage02.txt***» записываются товары, количество которых равно нулю,

– в другой текстовый файл с именем «***storage03.txt***» записываются оставшиеся товары, отсортированные по количеству (от большего к меньшему),

– по итогам записи файлов в консоль построчно выводятся наименования товаров, которых нет в наличии.

**Вариант №6**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл с именем «***eng01.txt***», содержащий произвольные англоязычные слова (по одному слову в строке, не более 40 слов в файле, длина каждого слова принудительно ограничена 80 символами), размещённый по пути «*../bin/Debug/*» решения (*solution*). Переписать слова в другой файл «***eng02.txt***», отсортировав их в алфавитном порядке. Результат, записанный в файл, вывести в консоль «***eng02.txt***», предусмотрев постраничный вывод.

Постраничный вывод организуется в условиях, при которых количество строк текста заведомо превышает количество доступных строк в окне консоли. Без необходимости перемещения прокруткой (скроллом). По заполнении доступных строк окна консоли текстом приостанавливать вывод до нажатия пользователем любой клавиши на клавиатуре. Выводить побуждающее сообщение вида: «Вывести следующие слова…». При подтверждении пользователем перехода к следующему набору слов очищать содержимое консоли.

**Вариант №7**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***bill.txt***», в который записать кассовый чек, в котором в одну длинную строку с разделителями вида «,» (запятая) последовательно записаны: код, наименование, количество единиц и цена за единицу товара. Файл разместить по пути «*../bin/Debug/*» решения (*solution*). Известно, что все поля чека обязательно заполнены и заполнены безошибочно. Учебная программа:

– выводит в консоль позиции чека в удобном для чтения формате;

– определяет общую стоимость покупки;

– формирует строку итого товарного чека с указанием общей стоимости покупки.

**Вариант №8**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл с именем «***store.txt***», в каждой строке которого записать: уникальный код, название, количество, стоимость товаров некоторой фирмы. Все параметры разделены между собой символом «*$*». Файл разместить по пути «*../bin/Debug/*» решения (*solution*). Создать три файла: «***names.txt***», «***values.txt***», «***prices.txt***», в каждый из который будут построчно записываться только наименования товаров, только количественные значения товаров, только стоимостные показатели товаров, соответственно. Сортировка записанных элементов в каждом файле – по возрастанию (от меньшего к большему).

**Вариант №9**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Программно создать текстовый файл «***integers.txt***». Файл разместить по пути «*../bin/Debug/*» решения (*solution*).Записать в файл ***n*** целых чисел, полученных псевдослучайным образом. Если сгенерированное значение содержит менее 5 знаков, то дополнить его спереди недостающим количеством нолей, если более 5 знаков – то записать последние пять знаков. В консоль при этом выводить неизменённые значения.

**Вариант №10**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить текстовый файл с именем «***reals.txt***». Файл разместить по пути «*../bin/Debug/*» решения (*solution*). В текстовом файле «***reals.txt***» разместить любые вещественные числа в любом количестве, но записанные по определённым правилам: разделителей между вещественными числами нет (отсутствуют даже пробелы), но известно, что каждое число записано с точностью до пяти знаков. Программно считать содержимое файла, вывести вещественные числа на экран построчно, указать после вывода количество чисел, а также указать их сумму.

**Вариант №11**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить текстовый файл «***rus01.txt***». Файл разместить по пути «*../bin/Debug/*» решения (*solution*). Наполнить текстовый файл произвольными словами на русском языке (с размещением по одному слову в строке, и не более, чем с сорока словами в файле, длина каждого слова ограничена пятьюдесятью символами). Программно переписать слова в другой файл с именем «***rus02.txt***», отсортировав их в обратном алфавитном порядке.

**Вариант №12**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить текстовый файл «***points01.txt***». Файл разместить по пути «*../bin/Debug/*» решения (*solution*). Текстовый файл наполнить координатами **точек**, разделённых между собой «;». **Координаты** точек перечислены через «,». Точки произвольные и не принадлежат никакой функциональной зависимости. Программно считать исходный текстовый файл и записать **точки** в новый файл «***points02.txt***» построчно, без разделителей в порядке увеличения их удалённости от начала координат. Разделитель **координат** тот же.

**Вариант №13**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить текстовый файл «***buy.txt***», в который записать произвольный кассовый чек, где в каждой строке последовательно разместить: код, наименование, количество единиц и цена за единицу товара. Известно, что все поля заполнены и заполнены безошибочно. В другой файл «***discount.txt***» последовательно записаны значения скидки, заданные в процентах и соответствующие им пороговые значения общей стоимости совершённой покупки. Оба файла разместить по пути «*../bin/Debug/*» решения (*solution*). Программно определить и вывести в консоль общую стоимость покупки с учётом скидки, а также указать далее получившуюся величину скидки.

**Вариант №14**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Взять любой текстовый файл и найти в нём все слова, которые в точности равны значению средней длины слова в этом файле. Сохранить найденные слова в переменную с разделителем «, », вывести в консоль и записать в том же виде в новый файл с именем «***avgresult.txt***». Файл разместить по пути «*../bin/Debug/*» решения (*solution*).

**Вариант №15**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). В коде программного обеспечения подготовить массив строк, содержащий цвета автомобильного светофора: «красный», «жёлтый», «зелёный». В результате работы программы записывать в файл «***lights.txt***» элементы массива построчно (каждый элемент в новой строке) трижды. Файл разместить по пути «*../bin/Debug/*» решения (*solution*). Выбор из массива очередного элемента для записи в файл регламентирован следующим правилом: если выпавшее вещественное псевдослучайное число меньше трети диапазона – выбирать нулевой элемент, если больше трети диапазона, но меньше двух третей – первый элемент, если вещественное псевдослучайное число попадает в остаток диапазона – второй элемент. По завершении заполнения файла считать его содержимое и вывести в консоль.

**Вариант №16**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***dots.txt***», содержащий координаты **точек**, разделённых между собой «;». **Координаты** перечислены через «,». Точки произвольные и не принадлежат никакой функциональной зависимости. Файл разместить по пути «*../bin/Debug/*» решения (*solution*). Считать исходный файл и записать **точки** в новый текстовый файл «***segments.txt***» построчно и попарно в порядке увеличения расстояний между парой точек, без каких-либо разделителей. Разделитель **координат** сохранить тем же. Задан алгоритм поиска: фиксируется первая точка в наборе; перебираются все последующие точки; для получающихся пар из зафиксированной и каждой последующей вычисляется длина отрезка; при обнаружении подходящей пары, дающей минимум длины отрезка, координаты записываются в файл; обе точки исключаются из дальнейшего рассмотрения; алгоритм повторяется. Алгоритм повторяется до тех пор, пока в наборе не останется ни одной точки или точка, к которой уже невозможно подобрать пару.

**Вариант №17**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***russian01.txt***», содержащий в себе слова, записанные на русском языке. Сформировать новый текстовый файл «***russian02.txt***», в котором будут записаны эти же слова с теми же разделителями между ними, но каждый символ слова будет закодирован его порядковым номером их таблицы *ASCII*. Файлы разместить по пути «*../bin/Debug/*» решения (*solution*). В качестве разделителя символов принять «|».

**Вариант №18**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***index.txt***», содержащий слова, перечисленные в нём через запятую. Файл разместить по пути «*../bin/Debug/*». Программно создать новый текстовый файл с гипертекстовой разметкой\* с тем же именем, но сохранив его с расширением «*html*» так, чтобы в браузере указанные слова отобразились в виде нумерованного списка. \*Подразумевается соответствующая модификация содержимого файла.

**Вариант №19**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***receipt01.txt***». Файл разместить по пути «*../bin/Debug/*». В файл записать кассовый чек, где в каждой строке последовательно записаны: код, наименование, количество единиц и цена за единицу товара. Известно, что все поля заполнены и заполнены безошибочно. Сформировать новый файл «***receipt02.txt***», в котором в том же формате будут записаны только те товары, которые окажутся дешевле указанной ценовой границы.

**Вариант №20**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***wds.txt***», содержащий некоторые, произвольные слова, перечисленные через запятую. Файл разместить по пути «*../bin/Debug/*». Создать в том же каталоге файлы с именами, соответствующими перебираемым в исходном файле словам. Разместить в каждом созданном файле последующее слово. Последний файл оставить пустым.

**Вариант №21**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***elements.txt***». Файл разместить по пути «*../bin/Debug/*». Файл содержит символы, целые, а также вещественные значения, перечисленные через пробел. Элементы собраны в файле произвольно, не обязательно в одной строке – это может быть и несколько строк не имеющего смысла (семантически незначимого) текста. По итогам разбора исходного файла сформировать три новых файла с именами «***integer.txt***», «***double.txt***» и «***char.txt***», в которых собрать данные соответствующих типов в порядке их упоминания в основном файле.

**Вариант №22**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***txtopers.txt***». Файл разместить по пути «*../bin/Debug/*». Файл содержит строки, в каждой из которых записана пара значений целого и/или вещественного типа (могут быть пары: как только целых, так и только вещественных), отделённых друг от друга пробелами и текстовыми надписями с действиями, например, «прибавить», «умножить», «разделить», «вычесть». В новый текстовый файл «***opersres.txt***» записать результаты выполнения указанных действий, разделённые пробелами. Для вещественных результатов сохранить только 5 знаков после плавающей запятой. Вещественные результаты с менее, чем 5 знаками, дополнить недостающими нулями.

**Вариант №23**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***traf.txt***». Файл разместить по пути «*../bin/Debug/*». В текстовый файл записаны целые значения в диапазоне от 1 до 120 включительно, разделённые между собой знаком «:». Составить учебную программу, которая считав из исходного файла последовательность, подскажет текущее действие для автолюбителя (водителя), записав это действие в новый файл «***dec.txt***» текстом. Известно, что целые значения в файле обозначают секунды. Красный сигнал не может гореть дольше полутора минут, жёлтый – не более 5 секунд, зелёный – дольше 40 секунд. Последовательность в исходном файле может содержать ошибки. В случае превышения времени на сигнал, остаток передаётся на следующий значимый сигнал (зелёный или красный). Превышения жёлтого сигнала срезаются до его верхней границы в пять секунд. Пользователь указывает сигнал начала последовательности вводом с клавиатуры. Во второй новый файл «***corr.txt***» попадает исправленная последовательность длительности сигналов. Количество элементов в этой последовательности может быть увеличено в результате правки, если выявлялись ошибки. При возникновении такой ситуации все значения сигналов последовательно дополняются до своих максимумов.

**Вариант №24**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***triple.txt***». В файле содержатся строки, состоящие только из трёх слов, разделённых между собой пробелами, в каждой. Файл разместить по пути «*../bin/Debug/*». Все слова обязательно имеют смысловую нагрузку и не превышают 10 символов каждое. Разбор файла выполняется по правилам:

1) сумма кодов символов (из таблицы *ASCII*) в первом слове – значение первого целочисленного операнда;

2) длина третьего слова – значение второго целочисленного операнда;

3) сумма кодов символов (из таблицы *ASCII*) второго слова, целочисленно делённая на длину второго слова, в зависимости от остатка от деления на 4 определяет операцию «умножение», «вычитание», «деление», «сложение» в указанном порядке. В новый текстовый файл «***cipher.txt***» записать результаты выполнения операций, разделённые пробелами. В случаях переполнения и возникновения прочих ошибок записывать в файл строковую константу «ошибка».

**Вариант №25**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***astr01.txt***» по строкам котором записано астрономическое время в формате «чч:мм:сс». Файл разместить по пути «*../bin/Debug/*». К каждому из значений прибавляется случайное целое количество секунд из диапазона от одной минуты до пяти часов. В новый тексовый файл «***astr02.txt***» записать в формате астрономического времени сгенерированное случайное значение, после которого через пробел указать, произошёл ли переход через сутки или нет. Для решения задачи не использовать тип данных *DateTime*, а также иные типы данных, связанные с датой/временем.

**Вариант №26**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). В программно генерируемом текстовом файле «***times01.txt***» построчно записана последовательность из двадцати псевдослучайных чисел, распределённых от «0» до «1», записанных с точностью до 19 знаков после плавающей запятой. Файл разместить по пути «*../bin/Debug/*». В новом программно генерируемом текстовом файле «***times02.txt***» указать астрономическое время в формате «чч:мм:сс», получаемое на основе данных, содержащихся в «***times01.txt***», если известно, что величине «0,0000115740740740741»\* соответствует одна секунда (00:00:01). \*Для тестирования удобно использовать *Microsoft Office Excel*.

**Вариант №27**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). В каждой строке программно создаваемого текстового файла (состоящего строго из 20 строк) с именем «***bins.txt***» содержатся набранные случайным образом бинарные последовательности из нолей и единиц, состоящие из 80 символов. Файл разместить по пути «*../bin/Debug/*». Известно, что первые восемь символов кодируют интенсивность канала красного цвета, следующие восемь символов – канал зелёного цвета, оставшиеся 8 символов – канал синего цвета. В новый файл «***colors\_27.txt***» записать в формате (*RRR*, *GGG*, *BBB*) десятичные значения цветовых каналов *R* – красный, *G* – зелёный, *B* – синий. Если полученное значение меньше 10 – дополнять его двумя нолями слева, если полученное значение больше 9, но меньше 100 – дополнять его одним нулём слева.

**Вариант №28**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Программно создать текстовый файл «***nums\_28.txt***» **.** Файл разместить по пути «*../bin/Debug/*». Программно записать в файл ***n*** (вводит пользователь с клавиатуры)вещественных чисел, полученных псевдослучайным образом из диапазона от 0 до 1. Если полученное значение оказывается меньше указанной пользователем точности ***eps*** (вводится с клавиатуры) – умножить его на 1000 и только после этого записать в файл.

**Вариант №29**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл «***elts.txt***». Файл разместить по пути «*../bin/Debug/*». Файл должен содержать символы, целые, а также вещественные значения, перечисленные через пробел. Содержимое может быть семантически незначимым (не иметь смысла). Элементы собраны в файле не обязательно в одной строке – это может быть несколько строк текста. По итогам разбора файла сформировать три новых файла с именами «***integer(…).txt***», «***double(…).txt***» и «***char(…).txt***», в которых собрать данные соответствующих типов с упорядочиванием их по возрастанию. В скобках имён файлов указать количество содержащихся в них элементов соответствующего типа.

**Вариант №30**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Взять любой текстовый файл (создать вручную, скачать в Интернет, найти существующий на персональном компьютере) и найти в нём размер самой длинной строки. Файл разместить по пути «*../bin/Debug/*». Если всё записано в одну строку, то размер самого длинного предложения. Если в файле не могут быть различены строки и/или предложения, то отказаться от использования такого файла при решении данной задачи. Сохранить в переменную только эту строку, вывести её в консоль, а в новый файл «***out\_30.txt***» записать инверсию первой половины данной строки и через символ «:» – вторую половину, записанную без изменений.

**Вариант №31**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Взять любой текстовый файл (создать файл вручную / скачать файл в Интернет / найти существующий файл на персональном компьютере), разместить его в каталоге решения (*solution*) «*Debug*», вложенном в каталог «*bin*», переименовать его в «***input\_31.txt***» и, обратившись к нему программно по его фиксированному имени, найти в нём длину наиболее протяжённой строки (если же всё содержимое текстового файла записано в нём в одну строку, то самого длинного предложения, завершающегося точкой «.»; если в текстовом файле не могут быть различены строки и/или предложения, то отказаться от использования такого файла при решении поставленной задачи). Сохранить в строковую переменную с именем «*maxString*» только самую длинную строку (самое длинное предложение), вывести содержимое переменной в консоль, а также записать в файл «***output\_31.txt***».

**Вариант №32**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Взять произвольный текстовый файл среди имеющихся на персональном компьютере или найденных в Интернет. В отчёте о проделанной работе, в разделе, посвящённом подготовке тестовых примеров, продемонстрировать способ получения файла и его содержимое. Файл разместить по пути «*../bin/Debug/*». Последовательно обнаруживать в файле кириллические символы. Создавать файлы с наименованиями символов без расширения (например, «а», «б», «в» и так далее). Внутри каждого такого файла размещать значение, равное общему количеству соответствующих символов в исходном файле. Предоставлять пользователю возможность ввода имени исходного текстового файла. Работа учебной программы должна начинаться с очистки каталога от файлов, полученных в результате предыдущего запуска программы (удалять все файлы без расширений).

**Вариант №33**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Подготовить исходный текстовый файл с именем «***m\_\_\_n.txt***», где ***m*** и ***n*** – целочисленные значения. Файл разместить по пути «*../bin/Debug/*». Файл содержит слова, перечисленные через запятую. Программно создать новый текстовый файл с именем «***result***», обозначить его как файл с гипертекстовой разметкой\*, сохранив с расширением «***html***». При открытии файла в браузере указанные слова должны отображаться в таблице размерности ***[m x n]***. Если слов в файле содержится больше, чем общее количество ячеек в таблице, то оставшиеся слова вывести через точку с запятой под таблицей в теге параграфа (абзаца). \*Подразумевается соответствующая модификация содержимого файла.

**Вариант №34**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* составить учебную консольную программу. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Составить учебную консольную программу, которая:

– считывает предложения из заранее подготовленного текстового файла «***input.txt***» (имя файла с расширением «*txt*» должно быть обязательно дополнено конструкцией «\_34») с заранее подготовленным содержимым,

– заменяет в предложениях все финализирующие знаки препинания («.», «!», «?», «…» и другие, а также их сочетания) на знак «%»,

– меняет заглавные буквы в началах предложений на строчные,

– записывает считанные из файла предложения в обратном порядке в текстовый файл «***output.txt***» (имя файла с расширением «*txt*» должно быть обязательно дополнено конструкцией «\_34»),

В процессе работы программы выводить в консоль сообщения о событиях, связанных заменами символов. Файлы разместить по пути «*../bin/Debug/*».

**Вариант №35**

В интегрированной среде разработки (*IDE*) *Microsoft Visual Studio* создать консольную программу для учебных целей. Убедиться, что *IDE* создала решение (*solution*) с классической структурой каталогов и файлов программного продукта, разрабатываемого на языке *Visual C#* (*csproj*). Создать в каталоге решения (*solution*) «*Debug*», вложенном в каталог «*bin*», текстовый файл с именем «***numbers01.txt»*** и записать в него вручную (не программно) простые числа из диапазона от 40 до 240 через доллар «*$*» (таблица простых чисел размещена в **Приложении**). Суть ручного заполнения заключается в том, чтобы не составлять дополнительный алгоритм расчёта простых чисел при выполнении данного задания. Составление программного обеспечения для расчёта простых чисел допускается, но оно не должно становиться частью подготавливаемого отчёта по данной работе. Считать содержимое файла «***numbers01.txt***» и записать в файл «***numbers02.txt***» каждое второе значение, записанное через процент «%». Все оставшиеся значения записать в файл «***numbers03.txt***» через диез «#». Новые файлы размещать в каталоге решения (*solution*) «*Debug*», вложенном в каталог «*bin*».

**Приложение**



1. Пример корректного подбора тестовых примеров в нужном количестве представлен в разделе 7 Сборника задач [Сафронов, А. И. Получение первичных профессиональных умений и навыков научно-исследовательской деятельности: Сборник задач для проведения аудиторных занятий по учебной практике / А. И. Сафронов, Н. Н. Зольникова, В. Г. Новиков. – Москва: РУТ. – 2019. – 91 с. – *EDN SXMWOD*]. Скачать можно на сайте Национальной Электронной Библиотеки (НЭБ) *elibrary.ru*: [*https://www.elibrary.ru/item.asp?id=46307421*](https://www.elibrary.ru/item.asp?id=46307421). [↑](#footnote-ref-1)
2. То же, что и сноска 1. [↑](#footnote-ref-2)
3. Пункт 6 на странице 391 [Сафронов, А. И. Способы проектирования эргономичных графических пользовательских интерфейсов для интеллектуальных транспортных систем / А. И. Сафронов // Интеллектуальные транспортные системы: Материалы III Международной научно-практической конференции, Москва, 30 мая 2024 года. – Москва: РУТ. – 2024. – С. 385-394. – *DOI 10.30932/9785002446094-2024-385-394*. – *EDN TGVERU*]. Скачать можно на сайте Национальной Электронной Библиотеки (НЭБ) *elibrary.ru*: [*https://www.elibrary.ru/item.asp?id=68533620*](https://www.elibrary.ru/item.asp?id=68533620). [↑](#footnote-ref-3)
4. Пример оформления отчётной документации [Сафронов, А. И. Составление отчётной документации по решённым задачам алгоритмизации и программирования: Учебно-методическое пособие для проведения аудиторных занятий по учебной практике / А. И. Сафронов, Н. Н. Зольникова, В. Г. Новиков. – Москва: РУТ. – 2018. – 83 с. – *EDN WNAIFP*]. Скачать можно на сайте Национальной Электронной Библиотеки (НЭБ) *elibrary.ru*: [*https://www.elibrary.ru/item.asp?id=46271697*](https://www.elibrary.ru/item.asp?id=46271697). [↑](#footnote-ref-4)
5. Правила изложены в разделе 8 того же учебно-методического пособия, что указано по сноске 4. [↑](#footnote-ref-5)
6. То же, что и сноска 1. [↑](#footnote-ref-6)
7. То же, что и сноска 1. [↑](#footnote-ref-7)
8. То же, что и сноска 3. [↑](#footnote-ref-8)