МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

М Е Т О Д И Ч Е С К И Е У К А З А Н И Я

к лабораторным работам

по программированию на языке Паскаль

ч а с т ь 3

(для студентов специальности 6.050102
"Компьютерная инженерия"
дневной формы обучения)

Рассмотрено
на заседании кафедры
КИ
Протокол № 1
от 31 августа 2010 г.

Утверждено
на заседании учебно-
издательского совета Дон-
НТУ
Протокол № 4
от 07 октября 2010 г.

Донецк ДонНТУ 2010

УДК 681.3(07)

Методические указания к лабораторным работам по программированию на языке Паскаль, часть 3 (для студентов специальност 6.050102 “Компьютерная инженерия” дневной формы обучения). Составители: В.И.Назаренко, К.Б.Юсупова, О.Ю.Чередникова. – Донецк: ДонНТУ, 2010– 49с.

Приведены методические указания к лабораторной работе №6, которая входит в цикл лабораторного практикума по программированию на языке Паскаль. В данной работе рассматриваются методы разработки алгоритмов и программ для задач обработки текстов. В методических указаниях приведены примеры использования различных процедур и функций для обработки строк, примеры выполнения задания в целом, а также 100 вариантов задач лабораторного практикума.

Составители: **доц. Назаренко В.И.**

асс. Юсупова К.Б.

асс. Чередникова О.Ю.

Ответственный
за выпуск проф. Святный В.А.

Рецензент доц. Федяев О.И.

Лабораторная работа №6

ОБРАБОТКА ТЕКСТА

Общие методические указания

Цель работы – практически освоить методы разработки алгоритмов и их программной реализации при обработке текстов, заданных в виде массива строк.

Во всех задачах, если это не оговорено дополнительно, производится обработка страницы текста, количество строк которой не превышает 50, а количество позиций в строке не превышает заданного значения (например, 66). Слова в тексте разделяются между собой одним или несколькими пробелами (или другими разделителями, если это указано в задаче). Перенос слов с одной строки на другую, как правило, не применяется (использование переноса слов заметно усложняет программу обработки текста).

В лабораторной работе предполагается, что входной текст записан во внешнем файле. После ввода из файла он отображается в виде массива строк. Преобразованный в соответствии с условием задачи текст размещается в том же массиве строк, что и входной (или в дополнительном массиве строк, если это требуется по алгоритму решения задачи), после чего он записывается в выходной файл. В некоторых случаях обработанные фрагменты входного текста непосредственно пересылаются в выходной файл, без образования массива строк. Если по условию задачи требуется формировать список любых элементов входного текста (слов, констант и т.п.), то элементы списка в выходном файле, как правило, должны разделяться запятой.

Если в процессе преобразования входного текста изменяются длины его строк, то перед записью текста в выходной файл нужно выполнить выравнивание строк текста путем переноса в свободную правую часть рассматриваемой строки некоторой начальной части следующей строки. Поскольку в заданиях к данной лабораторной работе перенос слов не используется, то в каждой строке текста должно быть размещено максимально возможное целое количество слов. Изменение промежутка между словами для полного выравнивания строк текста не выполняется, если это специально не указано в задаче.

В выходной файл выводятся также все другие результаты обработки входного текста. Печать результатов решения задачи должна выполняться путем чтения выходного файла и вывода прочитанных строк на принтер и (или) экран.

В каждой задаче входной файл должен быть описан как текстовый, а выходной – как типизированный, компонентами которого являются строки заданной длины (например, **`file of string[66]`**).

Если в задаче результаты работы программы непосредственно записываются в выходной файл, без формирования преобразованного массива строк, то перед записью в этот файл, как правило, должно быть обеспечено выравнивание текста по правой границе.

Отчет по лабораторной работе № 6

В состав отчета по лабораторной работе №6 должны входить:

- титульный лист;
- условие задачи;
- краткое описание разработанной программы;
- текст программы;
- результаты работы программы.

Текст программы должен печататься средствами компилятора Турбо Паскаля. Если по какой-либо причине для этого используется система Word, то в этом случае должен использоваться шрифт Courier New (в этом шрифте все символы имеют одинаковую ширину, что обеспечивает печать программы в структурированном виде). Для пп. 2 и 3 рекомендуется шрифт Times New Roman.

Результаты работы программы должны быть отпечатаны на принтере.

Примечание. Описание программы является обязательным. При его отсутствии отчет не принимается.

Операции, процедуры и функции для обработки строк

По отношению к строкам может быть использована лишь одна операция – операция сцепления (конкатенации), которая обозначается символом ‘+’. Если переменные `S1` и `S2` объявлены как строки, то выражение `S1+S2` означает, что к концу строки `S1` дописывается строка `S2`. При этом автоматически формируется соответствующее значение длины объединенной строки, которое записывается в ее нулевой байт.

Пример 1.

```
Var  S1,S2 : string[10];
        S3 : string[15];
        S4 : string;
Begin
  S1:='0123456789'; S2:='abcdefgh';
  S3:=S1+S2; S4:=S1+S2;
  Writeln('S3=',S3,'   S4=',S4);
  Будет напечатано:
    S3=0123456789abcde       { текущая длина S3 15 символов }
    S4=0123456789abcdefgh    { текущая длина S4 18 символов }
```

Предположим, что строку `S4` надо дополнить символами ‘*' до общей длины 25. Это можно сделать разными способами, два из которых приведены в примерах 2 и 3.

Пример 2.

```
Var  S4 : string;
Begin
  S4:='0123456789abcdefgh';
  While ord(S4[0])<=25 do      { ord(S4[0]) – текущая длина }
    S4:=S4+'*';                { строки S4 }
  Writeln('   S4 = ',S4);
```

Будет напечатано:

```
 S4 = 0123456789abcdefgh*******
```

Пример 3.

```
Var  i : byte;
    S4 : string;
Begin
  S4:='0123456789abcdefgh';
  For i:=ord(S4[0])+1 to 25 do
    S4[i]:='*';
  Writeln('   S4 = ',S4);
```

Будет напечатано:

```
 S4 = 0123456789abcdefgh
```

В примере 3 байты 21..25 строки на самом деле будут заполнены символом '*', но текущая длина строки, в отличие от примера 2, при этом не изменяется. Это объясняется тем, что в примере 2 обрабатывается в целом строка, а в примере 3 – лишь отдельные ее символы. Чтобы результат обработки в примере 3 был таким же, как и в примере 2, нужно после оператора **`For`** записать оператор `S4[0]:=chr(25)` (нулевой байт строки, как и

другие ее байты, воспринимается в программе как символ; приведенный выше оператор присваивания заносит в нулевой байт строки S4 символ, который соответствует порядковому номеру 25 таблицы ASCII).

Для строк допустимы все операции отношения. Сравнение строк выполняется слева направо по одному байту до тех пор, пока не будут обнаружены разные байты. Пусть в этих байтах размещены символы C1 и C2. Тогда

C1 < C2, если ord(C1) < ord(C2).

Если текущие длины сравниваемых строк неодинаковы, тогда более короткая строка дополняется символом chr(0), который меньше любого другого символа таблицы ASCII (символ chr(0) имеет порядковый номер 0 в таблице ASCII и условно обозначается Null; графического изображения этот символ не имеет).

Пример 4.

```
Var  b1,b2,b3 : boolean;
Begin
  b1:='abcdefgh'<'abcdxyzu';
  b2:='300'>'30';
  b3:=' '='1234567';
```

Здесь имеем b1 = true; b2 = true; b3 = false.

Строчные и прописные буквы в тексте считаются разными. В частности, 'd' ≠ 'D', так как ord('d') ≠ ord('D').

Для строк определен ряд процедур и функций.

Процедура **Delete(S,Start,n)** выполняет удаление n символов из строки S, начиная с позиции Start.

Пример 5.

```
Var  S : string[15];
Begin
  S:='0123456789';
  Delete(S,5,3);
```

Результат: S = '0123789'

Текущая длина строки S: исходная 10, конечная 7.

Процедура **Insert(S1,S2,Start)** - это вставка строки S1 в строку S2, начиная с позиции Start.

Пример 6.

```
Var  S1,S2,S3 : string[10];
Begin
  S1:='xyz'; S2:='01234'; S3:='0123456789';
  Insert(S1,S2,3); Insert(S1,S3,3);
```

Получим:

S2 = '01xyz234'; S3 = '01xyz23456';

Примечание. Для строки S3 объявлена длина 10 символов, поэтому после вставки строки S2 «лишние» символы '789' отсекаются (эти символы уже не помещаются в поле памяти переменной S3).

Процедура **Str(X,S)** выполняет преобразование числового значения арифметического выражения X и размещение результата в строке S. После выражения X можно записывать формат, аналогичный формату вывода в процедурах Write, Writeln. При этом автоматически производится округление в соответствии с заданным форматом дробной части числа.

Примечание. Выражением, в частности, может быть константа, переменная, функция.

Пример 7.

```
Var  x,y,z : real;
     k : integer;
     S1,S2,S3,S4 : string[10];
Begin
  x:=15.6789; y:=7; z:=123456789.123456789; k:=-789;
  Str(x:7:2,S1); Str(y:10,S2);
  Str(z:20:8,S3); Str(k:8,S4);
```

Получим:

S1 = ‘ 15.68’; S2 = ‘ 7.000E+00’;
S3 = ‘123456789.’ (*дробная часть числа z не поместилась в S3*);
S4 = ‘   -789’;

Для сравнения с процедурой Str рассмотрим работу процедуры Write при выводе числа в текстовый файл (на экран, принтер, диск). При размещении числа на экране или на бумаге каждая цифра (символ) числа занимает отдельную позицию. Следовательно, число при размещении его в текстовом файле – это строка.

Работа процедуры Write разделяется на два этапа:

- преобразование числовой переменной из ее внутренней машинной формы (в соответствии с описанием переменной в разделе **Var**) в строку;
- вывод строки на внешний носитель информации.

Первый этап работы процедуры Write эквивалентен работе процедуры Str.

Процедура **Val(S,X,Code)** выполняет преобразование строки S в число X. При этом в строке S не должно быть пробелов между цифрами числа, а форма записи символов должна соответствовать правилам записи числовых констант. Переменная Code типа integer – это код ошибки. Если преобразование завершилось успешно, то Code=0. В противном случае значение Code определяет позицию неверного символа, а переменной X при этом будет присвоено нулевое значение.

Пример 8.

```
Var  x1,x2,x3 : real;
     k1,k2,Code1,Code2,
     Code3,Code4,Code5 : integer;
     S1,S2,S3,S4,S5 : string[10];
Begin
  S1:=’  15.48’; S2:=’  15,48’; S3:=’  15.ab’;
  S4:=’  678’; S5:=’  6 78’;
  Val(S1,x1,Code1); Val(S2,x2,Code2);
  Val(S3,x3,Code3); Val(S4,k1,Code4);
  Val(S5,k2,Code5);
  Writeln(‘x1,Code1 = ‘,x1,’  ‘,Code1);
  Writeln(‘x2,Code2 = ‘,x2,’  ‘,Code2);
  Writeln(‘x3,Code3 = ‘,x3,’  ‘,Code3);
  Writeln(‘k1,Code4 = ‘,k1,’  ‘,Code4);
```

```
  Writeln('k2,Code5 = ',k2,'  ',Code5);
```

Будет напечатано:

```
x1,Code1 = 1.5480000000E+01  0
x2,Code2 = 0.0000000000E+00  5
x3,Code3 = 0.0000000000E+00  6
k1,Code4 = 678  0
k2,Code5 = 0  4
```

Для сравнения с процедурой `Val` рассмотрим работу процедуры `Read` при вводе числа из текстового файла (из клавиатуры или диска). Число в текстовом файле имеет вид строки, так как каждая цифра (символ) числа занимает отдельную позицию на внешнем носителе информации. Работу процедуры `Read`, как и процедуры `Write`, можно разделить на два этапа:

- преобразование строки в значение числовой переменной в соответствии с ее описанием в разделе **`Var`**;
- пересылка значения переменной в заданное поле памяти.

Первый этап работы процедуры `Read` эквивалентен работе процедуры `Val`.

Функция **`Copy(S,Start,n):string`** – это выделение из строки `S` подстроки длиной `n` байт, начиная с позиции `Start`.

Пример 9.

```
Var  S1,S2 : string;
Begin
  S1:='0123456789'; S2:=Copy(S1,3,4);
```

Получим:

```
S2 = '2345';
```

Функция **`Concat(S1,S2,…,Sn):string`** – это конкатенация (сцепление) строк `S1, S2,...,Sn`. Действие функции `Concat` эквивалентно последовательному выполнению операций сцепления `S1+S2+...+Sn`.

Функция **`Length(S):byte`** определяет текущую длину строки `S`. Действие этой функции эквивалентно вычислению выражений `ord(S[0])` или `byte(S[0])`.

Функция **`Pos(S1,S2):byte`** устанавливает позицию, в которой отмечается первое появление в строке `S2` подстроки `S1`. Если `S1` не содержится в `S2`, тогда выходное значение функции `Pos` равно нулю.

Пример 10.

```
Var  k1,k2,k3 : byte;
     S1,S2,S3 : string;
Begin
  S1:='abcdefghijklmnopq'; S2:='ghi'; S3:='gxi';
  k1:=Pos(S2,S1); k2:=Pos(S3,S1); k3:=Pos('k',S1);
```

Получим:

```
k1 = 7;  k2 = 0;  k3 = 11
```

Функция **`Upcase(ch):char`** выполняет преобразование строковой латинской буквы в прописную. Символы `ch` вне диапазона `'a'..'z'` остаются без изменения.

Пример 11.

```
Var  i : byte;
     S : string;
Begin
  S:='abc0123KLMN+фбщг-Uvwxyz';
  For i:=1 to length(S) do
    S[i]:=Upcase(S[i]);
```

Получим:

```
S = 'ABC0123KLMN+фбщг-UVWXYZ';
```

Для иллюстрации использования вышеуказанных процедур и функций рассмотрим несколько примеров.

Пример 12.

В строке задан текст. Слова текста разделены между собой одним или несколькими пробелами. В начале и в конце строки также могут быть пробелы. Сжать текст, удалив из него пробелы.

Рассмотрим три варианта программы.

```
Program Example12a;
Var  S : string;
     i : byte;
Begin
  Readln(S);
  For i:=1 to length(S) do
    If S[i]=' ' then
      Delete(S,i,1);
  Writeln(S);
End.
```

По поводу программы `Example12a` можно сделать следующие замечания:

1) Начальное и конечное значения параметра цикла **For** вычисляются до начала цикла и в процессе его выполнения не изменяются. Следовательно, уменьшение длины строки `length(S)` при `S[i]=' '` не отображается на количестве повторений цикла по параметру `i`. Это приводит к избыточной работе программы, но не отражается на правильности решения задачи.

2) Предположим, что в строке `S` имеются идущие подряд пробелы: `S[k]=' '` и `S[k+1]=' '`. При `i=k` будет удален элемент `S[k]`, а на его место будет переслан элемент `S[k+1]`, который также содержит пробел. Поскольку при новом повторении цикла **For** параметр цикла принимает значение `k+1`, то новый элемент `S[k]`, хотя он и включает в себя пробел, не будет анализироваться повторно и, как следствие, не будет удален из массива. Поэтому при наличии в массиве подряд идущих пробелов программа `Example12a` будет работать неверно.

Для правильного решения поставленной задачи нужно, чтобы программа после удаления элемента `S[k]=' '` повторно анализировала этот же элемент и переходила к рассмотрению элемента `S[k+1]` лишь при `S[k]<>' '`. Для этого нужно в программе вместо цикла **For** использовать цикл **While.**

```
Program Example12b;
Var  S : string;
     i : byte;
```

```
Begin
  Readln(S); i:=1;
  While i<= length(S) do
    If S[i]=' ' then
      Delete(S,i,1)
    Else
      Inc(i);
  Writeln(S);
End.
```

Правильная обработка строки S будет сделана также, если в цикле **For** выполнять просмотр символов строки не слева направо, а справа налево (программа Example12c).

```
Program Example12c;
Var  S : string;
     i : byte;
Begin
  Readln(S);
  For i:=length(S) downto 1 do
    If S[i]=' ' then
      Delete(S,i,1);
  Writeln(S);
End.
```

Пример 13. В строке задан текст. Слова текста разделены между собой одним или несколькими пробелами. В начале и в конце строки также могут быть пробелы. Определить количество слов в тексте и среднее количество букв в слове.

Предположим, что строка S имеет следующий вид (символ '-' здесь условно использован как пробел):

```
---abcde----fghijklmn----n-opq-xy--
```

Признаком начала слова является символ, отличный от пробела (непробел), признаком окончания слова – ближайший к нему пробел. Будем присваивать номера позиций, которые определяют положение слова в строке, переменным k1 и k2:

```
---abcde----fghijklmn----n-opq-xy--
   ↑    ↑    ↑         ↑
   k1   k2   k1        k2
```

Следовательно, при анализе строки в программе нужно многократно определять местоположение слова или, другими словами, позицию ближайшего пробела или непробела, начиная с заданной позиции k. Для выполнения такой работы целесообразно написать отдельные подпрограммы, оформив их в виде функции, поскольку выходом подпрограммы является одно значение – позиция пробела или непробела.

Поиск ближайшего пробела:

```
Function Space(S:string; k:byte):byte;
Var  i : byte;
Begin
  Space:=0;
```

```
  For i:=k to length(S) do
    If S[i]=' ' then
      Begin
        Space:=i; Exit
      End;
End { Space };
```

Если в строке S, начиная с позиции k, пробел не обнаружен, то выходное значение Space=0.

Поиск ближайшего непробела:

```
Function NotSpace(S:string; k:byte):byte;
Var  i : byte;
Begin
  NotSpace:=0;
  For i:=k to length(S) do
    If S[i]<>' ' then
      Begin
        NotSpace:=i; Exit
      End;
End { NotSpace };
```

Если в строке S после последнего символа последнего слова нет пробелов, то выходное значение функции NotSpace будет равно нулю.

Будем считать, что в программе Example13, которая реализует решение поставленной задачи, содержатся функции Space и NotSpace.

```
Program Example13;
Uses Crt;
Var  S : string;       { обрабатываемая строка }
     NumWords,         { количество слов }
     NumLetters,       { количество букв }
     k1,k2,            { левая и правая границы слова }
     len : byte;       { длина слова }
     MiddleLet : real; { среднее количество букв в слове }
     Cond : boolean;   { переменная для управления циклом }
{ ------------------------------ }
Функции  Space и NotSpace
{ ------------------------------ }
Begin
  ClrScr;
  Readln(S); Writeln('S = ',S);
  NumWords:=0; NumLetters:=0;
  MiddleLet:=0;
  k2:=0; Cond:=true;
  While Cond do
    Begin
      k1:=NotSpace(S,k2+1);
      If k1=0 then
        Cond:=false
      Else
        Begin
          k2:=Space(S,k1+1);
```

```
            If k2=0 then
              Begin
                k2:=length(S)+1; Cond:=false
              End;
            len:=k2-k1;
            Inc(NumWords); Inc(NumLetters,len);
          End;
      End;
    If NumWords>0 then          { блокировка деления на нуль }
      MiddleLet:=NumLetters/NumWords;
    Writeln('Количество слов = ',NumWords,
            '   Количество букв = ',NumLetters);
    Writeln('Средняя длина слова = ',MiddleLet:6:1);
End.
```

В программе осуществляется последовательное выделение слов текста путем определения границ `k1` и `k2`. Стартовое значение переменной `k2` принимается равным нулю. Цикл поиска слов продолжается до тех пор, пока является истинным значение переменной `Cond`.

В каждом цикле поиска определяются значения переменных `k1` и `k2`. Значение `k1` – это позиция ближайшего непробела, начиная с байта `k2+1` (в первом цикле `k2+1=1`). Если при обращении к функции `NotSpace` получено `k1=0`, то это означает, что до конца строки больше слов не обнаружено. Тогда переменной `Cond` присваивается значение `false`, что ведет к прекращению цикла поиска.

Если `k1<>0`, то определяется значение `k2`, то есть номер позиции ближайшего пробела, начиная с байта `k1+1`. Если при обращении к функции `Space` получено `k2=0`, то это означает, что до конца строки нет больше пробелов. Тогда правой границей слова считается байт `length(S)+1`. Вместе с тем значение `k2=0` указывает, что в тексте найдено последнее слово. Тогда переменной `Cond` назначается значение `false`, что после обработки этого слова вызовет прекращение работы цикла поиска.

Пример 14. В строке размещаются слова, которые состоят из русских букв, и целые десятичные числа. Перед числом может стоять знак '+' или '-'. Слова и числа разделены в тексте пробелами. Определить:

- количество чисел в тексте;
- общую сумму десятичных цифр, которые содержатся в этих числах.

Поиск числа в тексте выполняется аналогично поиску слова в предыдущем примере, но признаком начала числа является цифра или символы '+', '-'. Поэтому вместо функции `NotSpace` в данном случае для поиска числа будем использовать функцию `Number`.

```
Function Number(S:string; k:byte):byte;
Const  Sig = '+-0123456789';
Var  i : byte;
Begin
  Number:=0;
  For i:=k to length(S) do
    If Pos(S[i],Sig)>0 then
      Begin
        Number:=i; Exit
      End;
End { Number };
```

Будем считать, что в состав программы Example14 входят функции Space и Number.

```
Program Example14;
Uses Crt;
Var  S : string;       { обрабатываемая строка }
     i,                { параметр цикла }
     NumNumbers,       { количество чисел }
     k1,k2,            { левая и правая границы числа }
     Dig : byte;       { числовое значение цифры }
     Code : integer;   { код преобразования }
     Sum : longint;    { сумма цифр }
     Cond : boolean;   { переменная для управления циклом }
{ ------------------------------ }
Функции  Space и Number
{ ------------------------------ }
Begin
  ClrScr;
  Readln(S); Writeln('S = ',S);
  NumNumbers:=0; Sum:=0;
  k2:=0; Cond:=true;
  While Cond do
    Begin
      k1:=Number(S,k2+1);
      If k1=0 then
        Cond:=false
      Else
        Begin
          k2:=Space(S,k1+1);
          If k2=0 then
            Begin
              k2:=length(S)+1; Cond:=false
            End;
          Inc(NumNumbers);
          If (S[k1]='+') or (S[k1]='-') then
            Inc(k1);
          For i:=k1 to k2-1 do
            Begin
              Val(S[i],Dig,Code);
              Inc(Sum,Dig);
            End;
        End;
    End;
  Writeln('Количество чисел = ',NumNumbers,
          '   Сумма цифр = ',Sum);
End.
```

Пример 15. В состав текста входят слова и шестнадцатеричные числа, целая и дробная части которых разделены точкой. Признаком 16-го числа является префикс '$'. Перед числом может стоять знак '+' или '-'. Слова и числа разделены между собой одним или несколькими пробелами. Преобразовать шестнадцатеричные числа в десятичную систему счисления, изображая их в виде строки. В дробной части десятичного числа записывать не более чем 10 цифр. Незначащие нули, если они имеются в дробной части числа, удалить из состава строки.

Принцип поиска 16-го числа в основном аналогичен примеру 14, но в функции Number признаком начала числа является символ ‘$'. Границы числа, как и раньше, определяются переменными k1 и k2, при этом знак числа, если он есть, находится в позиции k1-1.

Дальше будем выполнять непосредственное преобразование найденного числа, записанного в строку Sb (без символа '$'):

```
Sb:=Copy(S,k1+1,k2-k1-1)
```

Обозначим через m длину строки Sb, через k – позицию точки, которая разделяет целую и дробную части числа:

```
m:=length(Sb);  k:=Pos('.',Sb)
```

Тогда цифры целой части занимают в строке Sb позиции с 1 до k-1, а дробной – с k+1 до m.

Пусть в целой части 16-го числа содержится n цифр. Тогда это число можно изобразить в следующем виде:

$$a_1 \cdot 16^{n-1} + a_2 \cdot 16^{n-2} + ... + a_{n-1} \cdot 16^1 + a_n \cdot 16^0$$

или по схеме Горнера:

$$(...(a_1 \cdot 16 + a_2) \cdot 16 + a_3) \cdot 16 + ... + a_{n-1}) \cdot 16 + a_n \ ,$$

где a_i, $i = 1..n$ - шестнадцатеричные цифры.

В приведенных ниже фрагментах программы предполагается, что входной текст размещается в строке S, из которой рассмотренным раньше методом определяется строка Sb, включающая в себя шестнадцатеричное число.

Программа преобразования целой части числа:

```
Const  Digs = '0123456789ABCDEF'; { шестнадцатеричные }
                                  { цифры }
Var  Nb1 : longint;  { целая часть десятичного числа }
     Nb2 : real;     { дробная часть такого числа }
     i,p,k,m : byte;
     S,              { входная строка }
     Sb,             { шестнадцатеричное число как строка }
     St1,            { целая часть 10-го числа как строка }
     St2 : string;   { то же для дробной части }
.................
  { Определение числа Sb из исходной строки S }
  m:=length(Sb); k:=Pos('.',Sb);
  Nb1:=0;
  For i:=1 to k-1 do
    Begin
      p:=Pos(Sb[i],Digs)-1;  { значение 16-ой цифры }
      Nb1:=Nb1*16+p
    End;
  Str(Nb1,St1);
  If S[k1-1]='+' then         { добавление знака числа }
    St1:='+'+St1
  Else
    If S[k1-1]='-' then
      St1:='-'+St1;
```

Предположим, что Sb = '1AE.5A'. Тогда
k = 4; m = 6; Nb1 = 430; St1 = '430'.

Пусть теперь в дробной части числа размещается p цифр. Это число можно изобразить в следующем виде:

$$b_1 \cdot 16^{-1} + b_2 \cdot 16^{-2} + b_3 \cdot 16^{-3} + ... + b_p \cdot 16^{-p} ,$$

где b_i, $i = 1..p$ - цифры дробной части числа.

Запишем это выражение в обратном порядке:

$$b_p \cdot 16^{-p} + b_{p-1} \cdot 16^{-(p-1)} + b_{p-2} \cdot 16^{-(p-2)} + ... + b_2 \cdot 16^{-2} + b_1 \cdot 16^{-1} .$$

Тогда по схеме Горнера

$$(...((b_p \cdot 16^{-1} + b_{p-1}) \cdot 16^{-1} + b_{p-2}) \cdot 16^{-1} + ... + b_2) \cdot 16^{-1} + b_1) \cdot 16^{-1}$$

или

$$(...((b_p / 16 + b_{p-1}) / 16 + b_{p-2}) / 16 + ... + b_2) / 16 + b_1) / 16$$

Считая, что в строке Sb дробная часть числа занимает позиции с `k+1` до `m`, получим представленную ниже программу преобразования дробной части числа:

```
Nb2:=0;
For i:=m downto k+1 do
  Begin
    p:=Pos(Sb[i],Digs)-1;
    Nb2:=(Nb2+p)/16;
  End;
Str(Nb2:12:10,St2);                { 1 }
While St2[length(St2)]='0' do
  Delete(St2,length(St2),1);       { 2 }
Delete(St2,1,1);                   { 3 }
```

Для `Sb = '1AE.5A'` получим:

- после { 1 } `Nb2 = 3.5156250000E-01;  St2 = '0.3515625000';`
- после { 2 } `St2 = '0.3515625';`
- после { 3 } `St2 = '.3515625'.`

Объединение целой и дробной частей числа:

```
St1:=St1+St2
```

Окончательный результат: `St1 = '430.3515625'.`

Пример 16. Поиск в строке пары слов, которые соответствуют заданному признаку.

Признаки могут быть разными (пара одинаковых слов, пара слов с обратным расположением букв и др.). Принципиальным здесь является организация цикла поиска.

Пусть в строке содержится *n* слов. В соответствии с условием задачи нужно сравнивать все пары слов:

```
1 - 2    1 - 3    1 - 4    1 - 5    1 - n
         2 - 3    2 - 4    2 - 5    2 - n
                  3 - 4    3 - 5    3 - n
                  ......................
                                (n - 1) - n
```

Если бы речь шла о сравнении элементов массива, цикл поиска можно было бы легко организовать с помощью оператора **`For:`**

```
For i:=1 to n-1 do
```

```
For j:=i+1 to n do
  If x[i]=x[j] then ...
```

Однако для строки заранее неизвестно количество записанных в ней слов. Поэтому вместо оператора **For** в данном случае нужно использовать оператор **While**. Суть организации цикла поиска можно проиллюстрировать таким фрагментом программы:

```
Cond1:=true; k2:=0;
While Cond1 do
  Begin
    k1:=NotSpace(S,k2+1); { Выделение первого из }
    If k1=0 then          { пары сравниваемых }
      Cond1:=false        { слов }
    Else
      Begin
        k2:=Space(S,k1+1);
        If k2=0 then      { Это последнее слово - после }
          Cond1:=false    { него нет других слов для }
        Else              { сравнения, поэтому цикл }
          Begin           { работу прекращает }
            Sb1:=Copy(S,k1,k2-k1);
            Cond2:=true; k4:=k2+1;
            While Cond2 do
              Begin
                k3:=NotSpace(S,k4+1); { Выделение второго }
                If k3=0 then          { из пары сравнива- }
                  Cond2:=false        { емых слов }
                Else
                  Begin
                    k4:=Space(S,k3+1);
                    If k4=0 then
                      Begin
                        k4:=length(S)+1; Cond2:=false
                      End;
                    Sb2:=Copy(S,k3,k4-k3);
                    { Анализ пары слов Sb1 и Sb2 }
                  End;
              End;
          End;
      End;
  End;
```

Пример выполнения задания

В качестве примера выполнения задания по лабораторной работе №6 приведены решения двух задач.

Условие задачи 1. В состав исходного текста входят слова, которые состоят из латинских букв, целые десятичные числа и разделители. Составить таблицу, в которой для каждой латинской буквы поставить в соответствие слово, в котором эта буква впервые встречается, и номер данного слова; после этого те слова, которые занесены в таблицу, удалить из исходного текста.

Решение задачи 1 приведено в представленной ниже программе `Labor6_1.`

Входной текст размещается в текстовом файле FileText, которому с помощью предописанной процедуры Assign ставится в соответствие вполне определенное имя внешнего файла.

Работа любой программы, в том числе и Labor6_1, должна быть проверена на системе тестов, в которых содержатся различные комбинации исходных данных. Если в процедуре Assign указать конкретное имя внешнего файла, например, 'Intext.txt', тогда при переходе от одного теста к другому необходимо предварительно набирать с клавиатуры новое содержимое файла 'Intext.txt' или пересылать в файл 'Intext.txt' другой файл, который содержит необходимый тест. Это связано со значительными неудобствами для пользователя.

Более приемлемым решением является такая организация работы программы, при которой пользователь мог бы указывать по запросу программы целиком или частично имя внешнего файла, в котором находятся необходимые исходные данные.

Процедура Assign(F,S) содержит два параметра: имя внутреннего файла F, описанное в разделе **Var**, и имя внешнего файла в виде строки S. Строка S может быть константой или переменной.

Имя внешнего входного файла в программе Labor6_1 формируется в строке NameInFile и может принимать значения InText0.txt, InText1.txt,..., InText9.txt, то есть всего допускается 10 файлов, которые содержат тестовые входные данные. Имена файлов отличаются лишь одной цифрой, которую условно можно считать номером входного файла. Этот номер пользователь сообщает с клавиатуры после соответствующего запроса программы. Ввод входного текста осуществляется из файла InTextP.txt, где P – номер входного файла. Переменная n в программе Labor6_1 определяет количество строк введенного текста. Введенный текст размещается в массиве строк St и выводится на экран дисплея, а также, если это нужно, печатается на принтере.

Таблица результатов состоит из четырех столбцов: порядковый номер строки таблицы; значение латинской буквы; номер слова, в котором впервые встречается эта буква (массив TabNumberWord); значение этого слова (массив TabWord). Всем элементам массива TabNumberWord задается начальное нулевое значение, а элементам массива строк TabWord - начальное пустое значение. Если при анализе входного текста не будет обнаружена какая-нибудь буква, то такие значения элементов массивов TabNumberWord и TabWord будут свидетельствует об отсутствии этой буквы.

В словах исходного текста могут быть как прописные, так и строчные латинские буквы, которые в данной задаче считаются равноценными. Для определения принадлежности очередного символа текста к буквам используется строка-константа AlphaBet, в которой размещены лишь прописные латинские буквы. В дальнейшем при анализе текста для преобразования строковых латинских букв в прописные используется функция UpCase.

Формирование таблицы результатов и одновременное удаление из текста слов, в которых впервые встретилась какая-нибудь буква, осуществляется в процедуре MakeTabResult.

При анализе текста последовательно определяются номер очередного слова NumberWord и количество впервые встреченных букв KolLetter. Этим переменным до начала просмотра текста присваивается нулевое значение.

Просмотр входного текста осуществляется в цикле с параметром i, где i определяет номер строки текста, то есть индекс элемента массива строк St.

Определение очередного слова в i-ой строке текста выполняется таким же образом, как это делается в примерах 13 и 14, но для индикации начала и конца слова используются функции SignBegin и SignEnd. Отделение слова, которое состоит из латинских букв, от десятичного числа осуществляется по его первому символу. После этого пересматриваются все буквы слова и, если очередная буква встречается в тексте впервые (TabNumberWord[k]=0, где k – номер буквы в строке AlphaBet), то она заносится в

таблицу результатов, а переменной CondDelete присваивается значение true. В последнем случае выполняется удаление данного слова из i-ой строки текста.

Следующий этап обработки текста – удаление избыточных пробелов, что выполняется процедурой DeleteSpaces. Здесь осуществляется удаление тех пробелов, после которых есть какой-нибудь разделитель, в частном случае другой пробел. Следовательно, в тексте сохраняются пробелы, которые записаны после других разделителей, например, после точки или запятой. После обработки очередной строки осуществляется удаление пробелов в начале и в конце строки (если они имеются).

В процессе просмотра исходного текста из его состава было удалено некоторое количество слов, что привело к сокращению длины строк текста. Для выравнивания этих строк по правой границе необходимо в каждую строку, длина которой меньше объявленной (в данном случае 66), перенести максимально возможную начальную часть следующей строки. Естественно, что при таком переносе целые десятичные числа, содержащиеся в составе исходного текста, не должны делиться на две части. Слова, в соответствии с общими методическими указаниями, также нельзя разделять знаком переноса. Поэтому из следующей строки надо взять такую начальную часть, которая заканчивается разделителем, а не буквой или цифрой (то есть брать в целом слова и числа).

Для определения принадлежности анализируемого символа к разделителям используется строка-константа Separs, описанная в разделе **Const.** В этой строке как разделители перечислены пробел, запятая, точка, точка с запятой, двоеточие. Для индикации начала и конца очередного слова, как и в процедуре MakeTabResult, используются функции SignBegin и SignEnd.

В процессе выравнивания текста не исключена возможность того, что общее количество строк n преобразованного текста сократится, если все символы последних строк будут перенесены в предыдущие строки. В связи с этим для перебора строк текста в процессе его выравнивания нельзя применять цикл **For.**

В процедуре RightBorder переменная i определяет номер строки обрабатываемого текста, is – номер строки нового (выравненного) текста. Перебор строк осуществляется в цикле **While** под управлением переменной CondText.

Для формирования новой строки используется переменная S1, в которую к началу цикла перебора заносится первая строка обрабатываемого текстового массива St. В переменную S2 заносится очередная i-ая строка обрабатываемого текста (начальное значение i=2). Если суммарная длина l1+l2 строк S1 и S2 меньше 66, то к строке S1 дописывается содержимое строки S2, а значение переменной i увеличивается на 1. Если при этом имеет место i>n, то управляющей переменной CondText присваивается значение false, что ведет к прекращению цикла перебора строк входного текста.

При l1+l2>66 осуществляется частичный перенос слов из строки S2 в строку S1, что выполняет цикл **While** под управлением переменной CondString. Границы очередного слова в строке S2 определяют переменные k1 и k2; значение l=k2-k1 – это длина слова. Если очередное слово может быть размещено в строке S1 (l1+l<=66), то оно записывается в S1 и удаляется из строки S2. Отношение l1+l>66 означает, что строка S1 уже заполнена. В этом случае цикл просмотра строки S2 прекращает свою работу, к значению переменной is добавляется 1, содержимое строки S1 записывается в строку St[is] нового текста, а в S1 записывается остаток строки S2, после чего номер строки i старого текста увеличивается на 1. При повторении внешнего цикла в S2 будет занесена очередная строка преобразованного массива St.

После окончания внешнего цикла **While** последней n-ой строке нового текста присваивается значение строки S1.

Преобразованный текст записывается в выходной типизированный файл FileOut, после чего читается из этого файла, выводится на экран и, если это указано пользователем, на принтер.

Примечание. В программе Labor6_1 для увеличения и уменьшения целой переменной на величину, обусловленную константой или второй целочисленной переменной, используются процедуры Inc (инкремент) и Dec (декремент).

```
Inc(i)      эквивалентно   i:=i+1;
Inc(i,5)    эквивалентно   i:=i+5;
Inc(i,k)    эквивалентно   i:=i+k;
Dec(i)      эквивалентно   i:=i-1;
Dec(i,5)    эквивалентно   i:=i-5;
Dec(i,k)    эквивалентно   i:=i-k.
```

Процедуры Inc и Dec порождают оптимизированный машинный код, что особенно полезно для больших программ, содействуя при этом уменьшению объема занимаемой памяти и увеличению быстродействия программы.

```
Program Labor6_1;
Uses Crt,Printer;
Const
    Nmax= 50;       { макс.количество эл-тов в массиве строк St }
    Enter = 13;     { код клавиши Enter }
    Escape = 27;    { код клавиши Esc }
    PressKey = 'Нажмите клавишу Enter';
    AlphaBet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'; { лат. алфавит }
    Separs = ' .,;:';         { список разделителей }
Type
    String66 = string[66];    { тип строки текста }
    StringAr = array[1..Nmax] of String66;
Var
    i,                        { параметр цикла }
    n,                        { кол-во строк в массиве St }
    NumberFile,               { номер входного файла }
    KolLetter                 { количество отмеченных букв }
          : byte;
    NumberWord
          : integer;          { текущий номер слова в тексте }
    IndPrinter : boolean;     { индикатор использования принтера }
    ch,                       { символ текста }
    Reply : char;             { символ ответа на запрос программы }
    S1,S2,                    { строки для преобразования текста }
    Sw,                       { очередное слово текста }
    NameInFile : String66;    { имя входного файла }
    St : StringAr;            { массив строк текста }
    TabWord                   { массив слов таблицы результатов }
       : array[1..26] of String66;
    TabNumberWord             { номера слов в таблице результатов }
       : array[1..26] of integer;
    FileText : text;          { входной файл }
    FileOut                   { выходной файл }
             : file of String66;
{ ------------------------------------------------------------ }
Procedure WaitEnter;
{ Задержка выполнения программы, пока не будет нажата }
{ клавиша Enter                                        }
Var ch : char;
Begin
```

```
  Repeat
    ch:=ReadKey;
  Until ord(ch) = Enter;
End { WaitEnter };
{ ------------------------------------------------------------ }
Procedure PrintString(X,Y:integer; S:string);
{ Вывод строки S в позицию X строки экрана с номером Y }
Begin
  GotoXY(X,Y);
  Write(S);
End { PrintString };
{ ------------------------------------------------------------ }
Procedure PrintKeyAndWaitEnter;
{ Вывод строки-константы PressKey в позицию 1 строки экрана 25 }
{ и задержка выполнения программы до нажима клавиши Enter      }
Begin
 PrintString(1,25,PressKey);
 WaitEnter;
 ClrScr;
End { PrintKeyAndWaitEnter };
{ ------------------------------------------------------------ }
Procedure ControlPageScreen(Var j,KeyExit:byte);
{ Контроль размера страницы на экране }
Const LengthPage=23;    { количество строк на странице }
      S='Следующая страница - Enter, конец просмотра - Esc';
Var ch : char;
Begin
  Inc(j); KeyExit:=0;
  If j=LengthPage then
    Begin
      j:=0;
      PrintString(1,25,S);
      Repeat
        ch:=ReadKey; KeyExit:=ord(ch);
      Until (KeyExit=Enter) or (KeyExit=Escape);
      ClrScr;
    End;
End { ControlPageScreen };
{ ------------------------------------------------------------ }
Procedure ScreenText(S:string);
{ Вывод на экран массива строк St }
Var  i : integer;
    j,KeyExit : byte;
Begin
  ClrScr; j:=0;
  Writeln(S);
  For i:=1 to n do
    Begin
      Writeln(St[i]);
      ControlPageScreen(j,KeyExit);
      If KeyExit=Escape then
        Exit;
    End;
  PrintKeyAndWaitEnter;
End { ScreenText };
```

```
{ ------------------------------------------------------------ }
Procedure PrinterText;
{ Печать на принтере входного текста }
Var  i : integer;
Begin
  Writeln(Lst);
  Writeln(Lst,'              В Х О Д Н О Й   Т Е К С Т');
  For i:=1 to n do
    Writeln(Lst,St[i]);
End { PrinterText };
{ ------------------------------------------------------------ }
Procedure ScreenTabResult;
{ Вывод на экран таблицы результатов }
Var  i : integer;
    j,KeyExit : byte;
Begin
  ClrScr; j:=1;
  Writeln('               Таблица результатов');
  For i:=1 to 26 do
    Begin
      Writeln(i:3,'   ',Copy(AlphaBet,i,1),'   ',
              'TabNumberWord[i]:3,'   ',TabWord[i]);
      ControlPageScreen(j,KeyExit);
      If KeyExit=Escape then
        Exit;
    End;
  PrintKeyAndWaitEnter;
End { ScreenTabResult };
{ ------------------------------------------------------------ }
Procedure ScreenOutFile;
{ Вывод на экран выходного файла }
Var j,KeyExit : byte;
Begin
  Seek(FileOut,0);
  ClrScr; j:=0;
  Writeln('             В Ы Х О Д Н О Й   Ф А Й Л');
  While not eof(FileOut) do
    Begin
      Read(FileOut,S1);
      Writeln(S1);
      ControlPageScreen(j,KeyExit);
      If KeyExit=Escape then
        Exit;
    End;
  PrintKeyAndWaitEnter;
End { ScreenOutFile };
{ ------------------------------------------------------------ }
Procedure PrinterOutFile;
{ Печать на принтере выходного файла }
Begin
  Seek(FileOut,0);
  Writeln(Lst); Writeln(Lst);
  Writeln(Lst,'              В Ы Х О Д Н О Й   Ф А Й Л');
  While not eof(FileOut) do
    Begin
```

```
      Read(FileOut,S1);
      Writeln(Lst,S1);
    End;
End { PrinterOutFile };
{ ------------------------------------------------------------ }
Function SignBegin(S:string66; k:byte):byte;
{ Поиск начала слова в строке S }
Var i : byte;
Begin
  SignBegin:=0;
  For i:=k to length(S) do
    If Pos(S[i],Separs)=0 then
      Begin
        SignBegin:=i; Exit
      End;
End { SignBegin };
{ ------------------------------------------------------------ }
Function SignEnd(S:string66; k:byte):byte;
{ Поиск конца слова в строке S }
Var i : byte;
Begin
  SignEnd:=0;
  For i:=k to length(S) do
    If Pos(S[i],Separs)>0 then
      Begin
        SignEnd:=i; Exit
      End;
End { SignEnd };
{ ------------------------------------------------------------ }
Procedure MakeTabResult;
{ Формирование таблицы результатов и удаление отмеченных слов }
Var  i,j,k,k1,k2 : byte;
     Cond,
     CondDelete : boolean;
Begin
  NumberWord:=0; KolLetter:=0;
  For i:=1 to n do
    Begin
      Cond:=true; k2:=0;
      While Cond do
        Begin
          k1:=SignBegin(St[i],k2+1);
          If k1=0 then
            Cond:=false
          Else
            Begin
              k2:=SignEnd(St[i],k1+1);
              If k2=0 then
                Begin
                  k2:=length(St[i])+1; Cond:=false;
                End;
              Sw:=Copy(St[i],k1,k2-k1);
              ch:=UpCase(Sw[1]);
              If Pos(ch,AlphaBet)>0 then
                Begin
```

```
                    Inc(NumberWord);
                    CondDelete:=false;
                    For j:=1 to length(Sw) do
                      Begin
                        ch:=UpCase(Sw[j]); k:=Pos(ch,AlphaBet);
                        If TabNumberWord[k]=0 then
                          Begin
                            TabNumberWord[k]:=NumberWord;
                            TabWord[k]:=Sw;
                            Inc(KolLetter);
                            CondDelete:=true;
                          End;
                      End;
                    If CondDelete then
                      Begin
                        Delete(St[i],k1,k2-k1);
                        k2:=k1;
                      End;
                    If KolLetter=26 then
                      Exit;
                  End;
              End;
          End;
      End;
End { MakeTabResult };
{ ------------------------------------------------------------ }
Procedure WriteTabResult;
{ Запись таблицы результатов в выходной файл }
Var  i : byte;
Begin
  ScreenTabResult;
  S1:=' ';
  Write(FileOut,S1);
  S1:='            ТАБЛИЦА ВЫХОДНЫХ РЕЗУЛЬТАТОВ';
  Write(FileOut,S1);
  S1:='-------------------------------------------------------';
  Write(FileOut,S1);
  S2:=': N п/п : Буква : Номер слова :        Слово         :';
  Write(FileOut,S2); Write(FileOut,S1);
  For i:=1 to 26 do
    Begin
      Str(i:2,S1); Str(TabNumberWord[i]:4,S2);
      S1:='   '+S1+'       '+Copy(AlphaBet,i,1)+'      '+
                S2+'        '+TabWord[i];
      Write(FileOut,S1);
    End;
End { WriteTabResult };
{ ------------------------------------------------------------ }
Procedure DeleteSpaces;
{ Удаление из текста избыточных пробелов }
Var  i,j : byte;
Begin
  For i:=1 to n do
    Begin
      S1:=St[i];
```

```
      j:=1;
      While j<length(S1) do
        If S1[j]=' ' then
          Begin
            ch:=S1[j+1];
            If Pos(ch,Separs)>0 then
              Delete(S1,j,1)
            Else
              Inc(j);
          End
        Else
         Inc(j);
      If S1[length(S1)]=' ' then
        Delete(S1,length(S1),1);
      If S1[1]=' ' then
        Delete(S1,1,1);
      St[i]:=S1;
    End;
End { DeleteSpaces };
{ ---------------------------------------------------------------- }
Procedure RightBorder;
{ Выравнивание скорректированного текста по правой границе }
Var  is,l,l1,l2,
     k,k1,k2 : byte;
     CondText,CondString : boolean;
Begin
  If n<2 then
    Exit;
  S1:=St[1]; l1:=length(S1);
  i:=2; is:=0; CondText:=true;
  While CondText do
    Begin
      S2:=St[i]; l2:=length(S2);
      If (l1+l2)<66 then
        Begin
          S1:=S1+' '+S2; l1:=length(S1);
          Inc(i);
          If i>n then
            CondText:=false
        End
      Else
        Begin
          k:=66-l1;
          If k>0 then
            Begin
              CondString:=true;
              While CondString do
                Begin
                  k1:=SignBegin(S2,1);
                  k2:=SignEnd(S2,k1+1);
                  If S2[k2]<>' ' then
                    Inc(k2);
                  l:=k2-k1;
                  If l1+l<=66 then
                    Begin
```

```
                    Sw:=Copy(S2,k1,k2-k1);
                    S1:=S1+' '+Sw;
                    Delete(S2,1,k2-1);
                  End
                Else
                  CondString:=false;
              End;
            Inc(is); St[is]:=S1;
            S1:=S2;
            If S1[1]=' ' then
              Delete(S1,1,1);
            l1:=length(S1);
            Inc(i);
            If i>n then
              CondText:=false;
          End;
      End;
  End;
  n:=is+1; St[n]:=S1;
End { RightBorder };
{ ------------------------------------------------------------ }

Begin

{ Установление соответствия между внутренним }
{ и внешним файлами }
  ClrScr;
  Writeln('Укажите номер входного файла (0..9)');
  Read(NumberFile); Str(NumberFile:1,S1);
  NameInFile:='InText'+S1+'.txt';
  Assign(FileText,NameInFile);
  Assign(FileOut,'OutText.dat');

{ Открытие используемых файлов }
  Reset(FileText);
  Rewrite(FileOut);

{ Запрос об использовании принтера }
  IndPrinter:=false;
  Writeln('Будет ли использован принтер ?  (Да,Нет)');
  Reply:=ReadKey;
  If Reply in ['Д','д','L','l'] then
    IndPrinter:=true;

{ Ввод и печать исходных данных }
  n:=0;
  While not eof(FileText) do
    Begin
      Inc(n);
      Readln(FileText,St[n]);
    End;
  ScreenText('Входной текст');
  If IndPrinter then
    PrinterText;
```

```
{ Установка таблицы результатов в начальное состояние }
  For i:=1 to 26 do
    Begin
      TabNumberWord[i]:=0;
      TabWord[i]:='';
    End;

{ Формирование таблицы результатов и удаление отмеченных слов }
  MakeTabResult;
  ScreenText('Текст после удаления отмеченных слов');

{ Запись таблицы результатов в выходный файл }
  WriteTabResult;

{ Удаление из текста избыточных пробелов }
  DeleteSpaces;
  ScreenText('Текст после удаления избыточных пробелов');

{ Выравнивание скорректированного текста по правой границе }
  RightBorder;
  ScreenText('Текст после выравнивания по правой границе');

{ Запись преобразованного текста в выходной файл }
  S1:=' ';
  Write(FileOut,S1);
  S1:='         П Е Р Е Т В О Р Е Н И Й    Т Е К С Т';
  Write(FileOut,S1);
  For i:=1 to n do
    Write(FileOut,St[i]);

{ Чтение и печать выходного файла }
  ScreenOutFile;
  If IndPrinter then
    PrinterOutFile;

{ Закрытие файлов }
  Close(FileText); Close(FileOut);

End.
```

```
                    И с х о д н ы й   т е к с т
ddfjh  5678  fjhiiii,AADCv .rty  8765  jhfb  rty  acd  vbnnm qwe
rrr vbnmnm  qwer,ttyui op 1234 ajdfj  hjkl  rxcv vbnm  pdfg rty  w
fghju 6789 890 hjk  fder vbmnd  nbnvbfdf  iyutzz dfg  n vcv cvvbi
adfgh yuiop. TybGF  JYbfbgn 9870 tty 12986  ghytn tRRew piuyt  nbg
nmmm  tyyui  ABCabc hgJG  tghnbgtyhbn  lkjhhttvbb  uyhfggb
ijhghfgw kknbgfghj,  ukjhg 654 oii  786,  lk7mnhgfgg

                Текст после удаления отмеченных слов
  5678  , .  8765    rty  acd
rrr vbnmnm  qwer,  1234 ajdfj     vbnm   rty  w
fghju 6789 890 hjk  fder vbmnd  nbnvbfdf   dfg  n vcv cvvbi
adfgh yuiop. TybGF  JYbfbgn 9870 tty 12986  ghytn tRRew piuyt  nbg
nmmm  tyyui  ABCabc hgJG  tghnbgtyhbn  lkjhhttvbb  uyhfggb
ijhghfgw kknbgfghj,  ukjhg 654 oii  786,  lk7mnhgfgg
```

Текст после удаления избыточных пробелов

```
5678,. 8765 rty acd
rrr vbnmnm qwer, 1234 ajdfj vbnm rty w
fghju 6789 890 hjk fder vbmnd nbnvbfdf dfg n vcv cvvbi
adfgh yuiop. TybGF JYbfbgn 9870 tty 12986 ghytn tRRew piuyt nbg
nmmm tyyui ABCabc hgJG tghnbgtyhbn lkjhhttvbb uyhfggb
ijhghfgw kknbgfghj, ukjhg 654 oii 786, lk7mnhgfgg
```

Текст после выравнивания по правой границе

```
5678,. 8765 rty acd rrr vbnmnm qwer, 1234 ajdfj vbnm rty w fghju 6
cvvbi adfgh yuiop. TybGF JYbfbgn 9870 tty 12986 ghytn tRRew piuyt
nbg nmmm tyyui ABCabc hgJG tghnbgtyhbn lkjhhttvbb uyhfggb ijhghfgw
kknbgfghj, ukjhg 654 oii 786, lk7mnhgfgg
```

ТАБЛИЦА ВЫХОДНЫХ РЕЗУЛЬТАТОВ

N п/п	Буква	Номер слова	Слово
1	A	3	AADCv
2	B	5	jhfb
3	C	3	AADCv
4	D	1	ddfjh
5	E	9	qwe
6	F	1	ddfjh
7	G	19	pdfg
8	H	1	ddfjh
9	I	2	fjhiiii
10	J	1	ddfjh
11	K	16	hjkl
12	L	16	hjkl
13	M	8	vbnnm
14	N	8	vbnnm
15	O	14	op
16	P	14	op
17	Q	9	qwe
18	R	4	rty
19	S	0	
20	T	4	rty
21	U	13	ttyui
22	V	3	AADCv
23	W	9	qwe
24	X	17	rxcv
25	Y	4	rty
26	Z	27	iyutzz

П Р Е О Б Р АЗ О В А Н Н Ы Й Т Е К С Т

```
5678,. 8765 rty acd rrr vbnmnm qwer, 1234 ajdfj vbnm rty w fghju 6
cvvbi adfgh yuiop. TybGF JYbfbgn 9870 tty 12986 ghytn tRRew piuyt
nbg nmmm tyyui ABCabc hgJG tghnbgtyhbn lkjhhttvbb uyhfggb ijhghfgw
kknbgfghj, ukjhg 654 oii 786, lk7mnhgfgg
```

Условие задачи 2. В состав входного текста входят слова и целые десятичные числа. Как разделители в тексте используются пробел, точка, запятая, точка с запятой, двоеточие, вопросительный и восклицательный знаки. Каждое число имеет не более чем девять десятичных цифр. Перед числом может стоять знак '+' или '-'. Требуется преобразовать числа в шестнадцатеричную систему счисления и записать их в выходной

файл, разделяя между собой запятыми. В качестве признака шестнадцатеричного числа использовать префикс $. При преобразовании чисел их знаки должны быть сохранены.

В составе программы Labor6_2 можно отметить два принципиальных момента: преобразование целого 10-го числа в 16-ю систему счисления и выравнивание строк выходного файла по правой границе. Первый из них достаточно ясен из текста процедуры Make16, на втором нужно остановиться отдельно.

В программе Labor6-1 выравнивание по правой границе выполняется после окончания формирования строк преобразованного текста, так как удаление отдельных элементов выполняется непосредственно в строках входного текста.

В программе Labor6_2 массив строк преобразованного текста не формируется, вместо этого формируются постепенно строки выходного файла по мере просмотра входного текста.

Очередная строка, которая в дальнейшем переносится в выходный файл, - это строка Sd. К началу обработки текста ей присваивается пустое значение, а ее длина len принимается равной нулю. После получения в строке Sb очередного 16-го числа производится проверка, можно ли строку Sb дописать к строке Sd (len+l<66). Если это подтверждается, то Sb дописывается в Sd с добавлением запятой для разделения со следующим элементом, при этом увеличивается соответствующим образом длина len. Если len+l>=66, то это означает, что очередное 16-е число Sb не может быть дописано в строку Sd. Тогда эта строка пересылается в выходной файл FileOut, а строке Sd присваивается значение строки Sb, то есть туда пересылается число, которое не могло разместиться в предшествующей строке выходного файла.

Пересылка строки Sd в выходной файл происходит лишь после полного заполнения этой строки, при этом после такой пересылки в Sd записывается очередное 16-е число. Следовательно, после окончания цикла **While** строка Sd не может быть пустой, в связи с чем она дополнительно пересылается в выходной файл. При этом предварительно в строке Sd удаляется последний символ, поскольку записывать запятую после последнего числа не имеет смысла.

```
Program Labor6_2;
Uses Crt,Printer;
Const
    Nmax= 50;       { макс.количество эл-тов в массиве строк St }
    Enter = 13;     { код клавиши Enter }
    Escape = 27;    { код клавиши Esc }
    PressKey = 'Нажмите клавишу Enter';
    Separs = ' .,;:?!'; { список разделителей }
Type
    String66 = string[66];   { тип строки текста }
    StringAr = array[1..Nmax] of String66;
Var
    i,                      { параметр цикла }
    n,                      { кол-во строк в массиве St }
    l,len,                  { длина строки }
    k1,k2                   { границы слова }
          : byte;
    Code : integer;         { признак преобразования }
    x : longint;            { число из текста }
    Cond,                   { управляющая переменная }
    IndPrinter : boolean;   { индикатор использования принтера }
    ch,                     { символ текста }
    Reply : char;           { символ ответа на запрос программы }
    S,                      { строка входного текста }
```

```
    Sb,                   { очередное слово текста }
    Sd : string66;        { строка для записи в выходной файл }
    St : StringAr;        { массив строк текста }
    FileText : text;      { входной файл }
    FileOut               { выходной файл }
             : file of String66;
{ ------------------------------------------------------------ }
Procedure WaitEnter;
{ Задержка выполнения программы, пока не будет нажата }
{ клавиша Enter                                          }
Var ch : char;
Begin
  Repeat
    ch:=ReadKey;
  Until ord(ch) = Enter;
End { WaitEnter };
{ ------------------------------------------------------------ }
Procedure ScreenText;
{ Вывод на экран массива строк St }
Var  i : integer;
Begin
  ClrScr;
  Writeln('              В Х О Д Н О Й  Т Е К С Т');
  For i:=1 to n do
    Writeln(St[i]);
End { ScreenText };
{ ------------------------------------------------------------ }
Procedure PrinterText;
{ Печать на принтере входного текста }
Var  i : integer;
Begin
  Writeln(Lst);
  Writeln(Lst,'              В Х О Д Н О Й  Т Е К С Т');
  For i:=1 to n do
    Writeln(Lst,St[i]);
End { PrinterText };
{ ------------------------------------------------------------ }
Procedure ScreenOutFile;
{ Вывод на экран выходного файла }
Var S : string66;
Begin
  Seek(FileOut,0);
  Writeln('              В Ы Х О Д Н О Й  Ф А Й Л');
  While not eof(FileOut) do
    Begin
      Read(FileOut,Sb);
      Writeln(Sb);
    End;
End { ScreenOutFile };
{ ------------------------------------------------------------ }
Procedure PrinterOutFile;
{ Печать на принтере выходного файла }
Begin
  Seek(FileOut,0);
  Writeln(Lst); Writeln(Lst);
```

```
  Writeln(Lst,'              В Ы Х О Д Н О Й   Ф А Й Л');
  While not eof(FileOut) do
    Begin
      Read(FileOut,S1);
      Writeln(Lst,S1);
    End;
End { PrinterOutFile };
{ ------------------------------------------------------------ }
Function SignBegin(S:string66; k:byte):byte;
{ Поиск начала слова в строке S }
Var i : byte;
Begin
  SignBegin:=0;
  For i:=k to length(S) do
    If Pos(S[i],Separs)=0 then
      Begin
        SignBegin:=i; Exit
      End;
End { SignBegin };
{ ------------------------------------------------------------ }
Function SignEnd(S:string66; k:byte):byte;
{ Поиск конца слова в строке S }
Var i : byte;
Begin
  SignEnd:=0;
  For i:=k to length(S) do
    If Pos(S[i],Separs)>0 then
      Begin
        SignEnd:=i; Exit
      End;
End { SignEnd };
------------------------------------------------------------ }
Function Make16(Sr:string66):string66;
{ Преобразование числа в 16-ую с/с }
Const  Dig16 : array[0..15] of char =
             ('0','1','2','3','4','5','6','7',
              '8','9','A','B','C','D','E','F');
Var     k : byte;
        d : char;
       Sm : string66;
Begin
  If (Sr[1]='+') or (Sr[1]='-') then     { преобразование строки }
    Val(Copy(Sr,2,length(Sr)-1),x,Code) { Sr в десятичное число }
  Else                                   { x типа longint }
    Val(Sr,x,Code);
  Sm:='';
  Repeat
    k:=x mod 16;    { формирование очередной 16-ой цифры k, }
                    { начиная с младшей цифры 16-го числа }
    x:=x div 16;
    d:=Dig16[k];    { преобразование цифры в символ и включение }
    Sm:=d+Sm;       { его в состав 16-го числа как строки }
  Until x=0;
  Sm:='$'+Sm;       { добавление префикса}
  If (Sr[1]='+') or (Sr[1]='-') then  { добавление знака }
```

```
    Sm:=Sr[1]+Sm;
  Make16:=Sm;
End { Make16 };
{ ------------------------------------------------------------ }
Begin

{ Установление соответствия между внутренним }
{ и внешним файлами }
  ClrScr;
  Assign(FileText,'InFile.txt');
  Assign(FileOut,'OutFile.dat');

{ Открытие используемых файлов }
  Reset(FileText); Rewrite(FileOut);

{ Запрос об использовании принтера }
  IndPrinter:=false;
  Writeln('Будет использоваться принтер ?  (Да,Нет)');
  Reply:=ReadKey;
  If Reply in ['Д','д','L','l'] then
    IndPrinter:=true;

{ Ввод и печать исходных данных }
  n:=0;
  While not eof(FileText) do
    Begin
      Inc(n);
      Readln(FileText,St[n]);
    End;
  ScreenText('Входной текст');
  If IndPrinter then
    PrinterText;

{ Поиск чисел в тексте и преобразование их в 16-ую с/с }
  Sd:=''; len:=0;
  For i:=1 to n do
    Begin
      S:=St[i];
      Cond:=true; k2:=0;
      While Cond do
        Begin
          k1:=SignBegin(k2+1);
          If k1=0 then
            Cond:=false
          Else
            Begin
              k2:=SignEnd(k1+1);
              If k2=0 then
                Begin
                  k2:=length(S)+1; Cond:=false
                End;
              Sb:=Copy(S,k1,k2-k1);
              Val(Sb,x,Code);
              If Code=0 then          { это число, а не слово }
                Begin
```

```
                    Sb:=Make16(Sb);
                    l:=length(Sb);
                    If len+l<66 then    { формирование строки }
                      Begin             { выходного файла }
                        Sd:=Sd+Sb+','; Inc(len,l+1);
                      End
                    Else
                      Begin
                        Write(FileOut,Sd);
                        Sd:=Sb+','; len:=l+1;
                      End;
                  End;
              End;
          End;
      End;
  Delete(Sd,length(Sd),1);   { удаление последней запятой }
  Write(FileOut,Sd);

{ Чтение и печать выходного файла }
  ScreenOutFile;
  If IndPrinter then
    PrinterOutFile;

{ Закрытие файлов }
  Close(FileText); Close(FileOut);
  WaitEnter;

End.
```

```
                    В Х О Д Н О Й  Т Е К С Т
abcdfr  8765   76,4567899;  iutrew  bnnn!  123456789  987654321
97531  13579,  jhgfdsiuytrreew?  24680  8642  rtyuhbjj   111111
555  7777777: 666666.   lkjhgfgdfsasasg  ijhgff ytrtre  3333333
99999999  8888888,,,,44444444 33333 222222222  111111111  uiytr
765765765  123123123  234234234  987987987  qwertyujh 765765765

                    В Ы Х О Д Н О Й  Ф А Й Л
$223D,$4C,$45B35B,$75BCD15,$3ADE68B1,$17CFB,$350B,$6068,$21C2,
$1B207,$22B,$76ADF1,$A2C2A,$32DCD5,$5F5E0FF,$87A238,$2A62B1C,
$8235,$D3ED78E,$69F6BC7,$2DA4A885,$756B5B3,$DF6217A,$3AE38013
```

Варианты заданий

В каждом варианте должен быть создан выходной типизированный файл, строки которого должны быть выравнены по правой границе. ***Преобразованный текст*** – это, как правило, текст, который читается из выходного файла.

---

1. Для разделения слов во входном тексте на русском языке используются пробел, точка, запятая, точка с запятой, двоеточие и тире, при этом любой из разделителей может быть окружен слева и (или) справа одним или несколькими пробелами. Перенести в выходной файл слова, которые начинаются и заканчиваются согласной буквой, кроме слов, которые имеют длину меньше трех литер. Слова в выходном файле разделять между собой запятой, после которой записывать один пробел.

---

2. Входной текст содержит список целых десятичных чисел, разделенных запятыми. Перенести в выходной файл числа-палиндромы, то есть числа, которые читаются одинаково слева направо и справа налево (например, 4554, 78987, 8 и т.п.). Если какое-либо из входных чисел содержит незначащие нули, то такие нули должны быть предварительно удалены. В выходном файле после разделяющей запятой записывать один пробел.

---

3. Слова исходного текста разделены между собой одним или несколькими пробелами. Требуется в каждой строке этого текста обменять местами первое и последнее слово с учетом того, что в общем случае длины первого и последнего слов могут быть различными. После этого все слова анализируемой строки нужно перенести в выходной файл, кроме тех слов, длина которых меньше трех символов. Слова в выходном файле разделять между собой запятыми, после запятой записывать пробел. Если в строке находится лишь одно слово, то, разумеется, никакой обмен не производится.

---

4. Русский текст содержит несколько чисел, записанных в римской системе счисления (диапазон чисел 1..3000). Перевести каждое число в десятичную систему счисления.

Примечание. В качестве признака римского числа можно принять то, что каждый символ в записи такого числа – это прописная латинская буква (I, V, X, L, C, D, M). При определении числа применяется такое правило: если данная римская цифра меньше следующей, то ее значение отнимается от числа, в противном случае – добавляется. Например, для числа MMCMXLVIII получим 1000+1000-100+1000-10+50+5+1+1+1 = 2948.

---

5. Входной текст содержит два предложения, каждое из которых заканчивается точкой. Предложение может занимать несколько строк, причем в последней строке первого предложения может начинаться второе предложение. Определить набор букв, которые должны быть добавлены к первому предложению, чтобы из его состава можно было сконструировать второе предложение.

---

6. В каждом из слов входного текста сгруппировать повторяемые символы в том порядке, в котором они встречаются в данном слове. Например, для слова

'AX56KLDFA5XPUYK5XZY67DFUYPAJG765VRSZ'

получим

'AAAXXX5555666KKLDDFFPPUUYYYZZ77JGVRS'.

Если слово создано лишь одним символом (например, aAAaaAaa), то такое слово удалить из состава текста.

---

7. Каждая строка входного текста – это беспрерывная последовательность цифр. Каждые две цифры - это порядковый номер буквы русского алфавита; код, который равняется 00 или превышает 32, считается кодом пробела. Выполнить декодирование входного текста, заменив каждое двузначное число соответствующей буквой. Если какая-нибудь строка содержит нечетное количество цифр, то вставить перед последним символом цифру '0'.

---

8. Входной текст состоит из русских букв, цифр и символов-разделителей (пробел, точка, запятая и т.п.). Для записи слов используются прописные и строчные буквы, считающиеся эквивалентными. Определить в абсолютных и относительных единицах частоту каждой буквы и записать в выходной файл в виде таблицы полученные результаты в порядке уменьшения относительной частоты букв. Как относительную единицу измерения использовать проценты, при этом записывать их в таблицу с двумя цифрами в дробной части. Слова, которые включают в себя букву с наименьшей, но не нулевой, относительной частотой, удалить из состава текста.

---

9. Входной текст содержит список идентификаторов. Элементы списка разделены между собой запятыми, слева и (или) справа от запятой могут быть один или несколько пробелов. В составе идентификатора могут быть как прописные, так и строчные буквы. Перенести в преобразованный текст список ошибочных идентификаторов, разделив их между собой запятыми.

Примечание. Идентификатор состоит из латинских букв, цифр и знака подчеркивания, при этом начинаться он должен с буквы. Знак подчеркивания не может быть последним, после этого знака должна быть буква или цифра.

---

10. Входной текст содержит слова, которые состоят из строчных латинских букв, целые десятичные числа и разделители (пробел, запятая, точка и т.п.). В состав слова могут входить цифры, но в этом случае первым символом должна быть буква. Удалить из текста все слова с нечетными порядковыми номерами, если длина этих слов не превышает 5 символов. В словах с четными порядковыми номерами переставить буквы в обратном порядке, если эти слова не имеют в своем составе цифр. Нумерации подлежат лишь слова, а не числа.

---

11. Слова входного текста состоят из прописных и строчных латинских букв. В каждом слове расположить по алфавиту буквы, которые входят в его состав, считая при этом, что прописные и строчные буквы эквивалентны. Например, для слова

'AHGgfraUYTEuerfSTssASGgXYzxyZX'

получим

'AaAEeffGgGgHrrSssSTTUuXxXYYyz'.

Если в слове все буквы одинаковые и его длина не превышает 5 символов, то удалить такое слово из текста.

---

12. Слова в тексте разделены между собой одним из таких символов: пробел, точка, запятая, точка с запятой и двоеточие, при этом любой из этих разделителей может быть окружен слева и (или) справа одним или несколькими пробелами. Определить среднюю длину *S* слов входного текста и перенести в преобразованный текст слова, длина которых не превышает значения *S*. Слова в новом тексте разделять между собой запятой с пробелом.

---

13. Входной текст содержит слова на русском языке и целые четверичные числа. Требуется преобразовать эти числа в десятичную систему счисления, после чего переставить их цифры в обратном порядке. Преобразованные числа не должны содержать незначащих нулей. К словам, которые имеют две или три буквы, дописать в их конец эти же буквы, но в обратном порядке.

---

14. Входной текст содержит слова на русском языке. Для записи слов используются как прописные, так и строчные буквы, которые считаются эквивалентными. Разделителями в тексте являются пробел, точка, запятая, точка с запятой и двоеточие, при этом любой из них может быть окружен слева и (или) справа одним или несколькими пробелами. Перенести в преобразованный текст слова входного текста, которые содержат удвоенные гласные (но не слова, в которых имеется группа из трех или более одинаковых гласных, которые расположены подряд). Слова в этом тексте разделять запятой с пробелом, каждое слово должно начинаться с прописной буквы, все другие буквы в слове должны быть строчными.

---

15. Текст содержит слова и целые десятичные числа, разделенные между собою одним или несколькими пробелами. Слова формируются из строчных латинских букв. В состав слов могут входить также цифры, но в этом случае первым символом является буква. Перед числом может стоять знак ‘+' или '-'. В выходной файл записать сначала слова, сгруппированные по алфавиту, а потом числа, сгруппированные по уменьшению количества их цифр. При этом в выходной файл переносить лишь слова, которые не имеют

в своем составе цифр, и числа без знака. Считать, что как слова, так и числа не могут иметь больше чем 15 символов.

Примечание. Рекомендуется сначала создать массив слов и массив чисел, сгруппировать их в соответствии с условием задачи, а потом перенести в выходной файл.

---

16. Входной текст содержит список арифметических констант в форме десятичных чисел с плавающей запятой. Элементы списка разделены между собой одним или несколькими пробелами. Перенести в преобразованный текст список констант, которые превышают по модулю допустимое для типа real значение (1E38). Элементы списка в выходном файле разделить между собой запятыми, а после каждой запятой записать один пробел. Для анализа констант процедуры `Val` и `Str` не использовать. Например, для входного списка

1.2E+12 -43.677E37 2567.234E+34 0.00087E45 14.6E-5 -6.6E+38 777E101

получим -43.677E37, 2567.234E+34, 0.00087E45, -6.6E+38, 777E101.

---

17. В тексте содержатся слова на русском языке, разделенные между собою пробелом, запятой или точкой. Для записи слов используются как строчные, так и прописные буквы, считающиеся эквивалентными. Перенести в выходной файл слова, которые начинаются с гласной буквы, переставив при этом буквы каждого слова в обратном порядке. Слова в выходном файле должны быть разделены между собой одним пробелом.

---

18. Слова входного текста разделены между собой одним или несколькими пробелами. Среди символов слова могут быть как буквы, так и цифры, но первой всегда записывается буква. Анализируя текст в целом, выполнить перестановку его слов в обратном порядке. Если среди символов слова есть хоть одна цифра, удалить такое слово из состава текста. В преобразованном тексте слова должны быть разделены между собой лишь одним пробелом. После преобразования текста его следует переписать в выходной файл.

Указание. Сначала выполнить перестановку слов в каждой строке, а потом переставить в обратном порядке компоненты массива строк.

---

19. Входной текст содержит список десятичных констант в форме чисел с плавающей запятой. Элементы списка разделены между собой одним или несколькими пробелами. Записать эти константы в выходном файле в виде целой и дробной частей, разделенных точкой. Если при этом общее количество цифр в преобразованной константе больше 15, то ее в выходной файл не включать. Элементы выходного файла разделять между собой запятой, после запятой записывать один пробел. Например, для входного списка

35.31E8 35.31E20 +35.31E-8 35.31E-20 123456789.987654321E+0-0.5E-3 1.48E+01

получим 3531000000.0, +0.0000003531, -0.0005, 14.8.

---

20. Входной текст содержит несколько арифметических выражений, в состав которых входят идентификаторы простых переменных, целые десятичные числа, знаки арифметических операций и круглые скобки. Выражения разделены между собой точкой с запятой. Между элементами выражений могут быть пробелы. Каждое выражение размещается в пределах одной строки, продолжение выражения на следующей строке не производится. Проанализировать в каждом выражении баланс открывающих и закрывающих скобок и в случае его нарушения удалить избыточные внешние скобки. Удалить также пробелы между элементами выражений. В преобразованном тексте после каждой точки с запятой записать один пробел.

При записи преобразованного текста в выходной файл с целью выравнивания строк по правой границе допускается перенос части выражения на следующую строку. При этом

никакого знака переноса не ставится, но имена переменных и числа, которые входят в выражение, не должны разрываться.

---

21. Для разделения слов во входном тексте на русском языке используются пробел, точка, запятая, точка с запятой, двоеточие и тире, при этом любой из разделителей может быть окружен слева и (или) справа одним или несколькими пробелами. Для записи слов используются как строчные, так и прописные буквы, считающиеся эквивалентными. Перенести в выходной файл из входного текста пары слов, которые состоят из одинаковых букв (например, товаР и авТор, Салат и Атлас и т.п.). Между словами одной пары ставить тире, окруженное слева и справа одним пробелом, пары слов отделять одну от другой запятой с пробелом. Для записи слов в выходном файле использовать лишь строчные буквы. Пример: товар - автор, салат – атлас.

Примечание. Условие задачи означает, что в любое из пары слов должны входить не только одинаковые буквы, но и их количество также должно быть одинаковым. Например, пары слов 'абатка' и 'аббата ', ‘калоша’ и ‘лошока’ не отвечают заданному условию.

---

22. Текст на русском языке содержит список украинских и русских фамилий мужского рода. Элементы списка разделены между собою одним или несколькими пробелами. Преобразовать список таким образом, чтобы сначала шли все фамилии с окончанием -ко, потом – фамилии с окончанием -ин, в третьей группе – фамилии с окончанием -ов, в последней группе – все другие фамилии. В выходной файл фамилии с окончанием –ук и -юк не пересылать. Относительная последовательность фамилий в каждой группе должна быть такой же, как и во входном списке. Элементы нового списка разделять между собой запятой с пробелом.

Рекомендация. Создать 4 буферных файла, а потом объединить их в один выходной файл, уничтожив буферные файлы, которые стали после этого ненужными.

---

23. В тексте, записанном латинскими буквами, могут быть группы слов, заключенные в круглые скобки. Каждая группа записывется в пределах одной строки, без переноса на следующую. Между словами текста в качестве разделителей могут быть пробел, запятая или точка, при этом любой из них может быть окружен одним или несколькими пробелами. При формировании преобразованного текста удалить из него указанные группы слов (вместе со скобками).

---

24. Слова текста разделены пробелами. Каждое слово – это последовательность алфавитно-цифровых символов, например, A2B5E33426XRVT0Z11U. Эту входную последовательность нужно интерпретировать таким образом: если очередной символ – цифра n (n = 0..9), то это рассматривается как указание об n повторениях следующего символа независимо от того, является ли этот символ цифрой или произвольной буквой; при этом повторяемый символ больше не рассматривается как возможный коэффициент повторения. Для приведенного выше примера имеем

ABBEEEEE3332222XXXXXXRVT1U

Нужно все слова текста преобразовать по описанному выше правилу и записать их в выходной файл. Слова в выходном файле должны разделяться между собой запятой с пробелом.

---

25. Во входном тексте беспрерывным потоком записаны группы латинских букв и группы цифр, причем длина каждой группы не превышает 10 символов. При этом любая очередная строка считается продолжением предыдущей строки (но знак переноса не ставится). Отделить каждую группу одним пробелом от соседних групп. В преобразованном тексте переноса элементов на следующую строку не должно быть. Если последняя группа рассматриваемой строки и первая группа следующей строки относятся к одному типу (цифры или буквы), то объединить их в одну группу при условии, что

суммарная длина такой группы не превышает 10 символов. Определить общее количество групп цифр и количество групп букв.

Вполне очевидно, что пробел вставляетс, если два смежных символа принадлежат к разным группам (цифра и нецифра или наоборот). Пробел после последнего символа строки записывается, если первый символ следующей строки относится к другой группе.

---

26. Входной текст содержит слова на русском языке и целые десятичные числа, разделенные между собой одним или несколькими пробелами. Для записи слов используются как строчные, так и прописные буквы. В каждом числе переставить его цифры в обратном порядке. Удалить из числа незначащие нули, если они образовались после перестановки цифр. В словах текста заменить прописные буквы строчными, за исключением первой буквы слова, если она прописная. Как разделитель в преобразованном тексте использовать лишь один пробел.

---

27. Во входном тексте используются только латинские буквы и разделители (пробел, запятая, точка, точка с запятой). Сгруппировать слова этого текста в порядке увеличения относительного количества в них гласных букв. Считать, что длина отдельного слова не превышает 20 символов. Слова в новом тексте разделять между собой запятой с пробелом. Как единицу относительного количества использовать проценты, дробная часть которых имеет две цифры.

---

28. Для разделения слов во входном тексте используются пробел, точка, запятая, точка с запятой, двоеточие и тире, при этом любой из разделителей может быть окружен слева и (или) справа одним или несколькими пробелами. Перенести в выходной файл из входного текста слова, в которых все буквы разные. Если слово имеет хотя бы одну цифру, то оно переносу не подлежит. В преобразованном тексте разделять слова лишь одним пробелом.

---

29. Входной текст содержит список шестнадцатеричних чисел, разделенных между собой запятыми. Целая и дробная часть каждого числа разделены точкой. Переставить цифры числа в обратном порядке, отдельно для целой и дробной частей. Удалить незначащие нули, если они появились при перестановке цифр. Если в дробной части числа нет значащих цифр, то удалить из состава числа также разделяющую точку. После выполнения указанных действий преобразовать числа, содержащиеся в тексте, в десятичную систему счисления, при этом дробная часть числа должна иметь не больше чем 4 цифры. В выходном файле числа должны быть разделены между собой запятой с пробелом.

---

30. Во входном тексте задан список десятичных констант в форме целой и дробной частей, разделенных точкой. Элементы списка разделены между собой одним или несколькими пробелами. Записать эти константы в форме с плавающей запятой, предусмотрев в целой части мантиссы одну значащую цифру (не нуль). В записи порядка числа обязательно указывать после буквы ‘E' знак порядка '+' или '-'. Значение порядка записывать с двумя цифрами.

Примечание. Исключением является число 0, оно изображается в виде 0.0E+00.

---

31. Входной текст содержит слова, которые состоят из латинских букв, числа, знаки арифметических операций ('+', '-', '*', '/'), разделительные знаки (запятая, точка и др.) и скобки. Сформировать новый текст, записав в него сплошным потоком три группы символов (буквы, цифры, другие символы), придерживаясь при этом их входной последовательности. Пробелы в новый текст не переносить. Любая из упомянутых выше групп символов может занимать больше одной строки. Группы разделить между собою точкой с запятой, после которой записать один пробел.

Рекомендация. Трижды просматривать входной текст, перенося при этом в выходной файл соответствующие символы.

---

32. Слова текста формируются латинскими буквами, как разделители слов используются пробел, запятая и точка, при этом любой из них может быть окружен одним или несколькими пробелами. Буквы, из которых складываются слова входного текста, сгруппировать в новые слова по *k* литер в каждом (переменную *k* ввести с клавиатуры, предусмотрев ее значение в диапазоне 1..10). При формировании новых слов необходимо сохранить входную относительную последовательность букв. Каждая строка входного текста считается продолжением предыдущей строки. Слова нового текста разделять между собой запятой с пробелом.

Например, для входного текста

Fgrty, uytkjhg treLKhfdui, nhSDgjkll.hjgkj.jhkliom,hjkkll

при $k=3$ получим

Fgr, tyu, ytk, jhg, tre, LKh, fdu, inh, SDg, jkl, lhj, gkj, jhk, lio, mhj, kkl, l

Рекомендация. Удалить из входного текста все разделители, после чего группировать его символы в соответствии с условием задачи.

---

33. Входной текст содержит несколько арифметических выражений, в состав которых входят идентификаторы простых переменных, целые десятичные числа, знаки арифметических операций и круглые скобки. Выражения разделены между собой точкой с запятой. Между элементами выражений могут быть пробелы. Перенести в преобразованный текст отдельно для каждого выражения список неповторяемых в нем констант. Константы в новом тексте разделять между собой запятыми, а группы констант, которые относятся к отдельному выражению – точкой с запятой. После точки с запятой записывать один пробел.

Пример. Для строки

```
5*a+(12b-5)+1; St-7R+11/(7-R); 14-3n+14
```

получим

```
5,12,1; 7,11; 14,3
```

---

34. В тексте беспрерывным потоком расположены цифры и другие символы, при этом каждая строка не считается продолжением предыдущей строки. Перенести в преобразованный текст все группы цифр, которые идут подряд, расположив их в порядке уменьшения длины группы. Считать, что длина беспрерывной группы цифр не превышает 20 символов. Элементы преобразованного текста разделять между собой запятой с пробелом.

Рекомендация.

Просматривая текст, создать массив цифровых групп, после чего отсортировать его в порядке уменьшения длины группы и переписать в выходной файл. Если две или более групп имеют одинаковую длину, то расположить их по уменьшению числового значения. При этом нет смысла явно преобразовывать строку в число с помощью процедуры Val, для этого достаточно просто сравнивать между собою строки.

---

35. В качестве разделителей в словах входного текста используются пробел, запятая, точка и точка с запятой, при этом любой из них может быть окружен одним или несколькими пробелами. Перенести в выходной файл слова, которые состоят из тех же букв, что и первое слово текста (не учитывая при этом количество повторений каждой буквы). Первое слово входного текста также должно быть записано в выходной файл. Слова в преобразованном тексте разделять запятой с пробелом.

---

36. Входной текст содержит слова, которые состоят из строчных и прописных латинских букв, целые десятичные числа и разделители (пробел, запятая, точка с запятой, двоеточие). Удалить из каждой строки текста слова, которые уже раньше встречались в

этой же строке. При этом прописные и строчные буквы считаются эквивалентными. В оставшихся словах первая буква должна быть прописной, остальные – строчными. Слова в преобразованном тексте разделять между собою лишь одним пробелом.

Примечание. В таблице ASCII строчные латинские буквы расположены через 32 позиции после прописных. Следовательно, 'a'=chr(ord('A')+32), 'b'=chr(ord('B')+32) и т.д.

Рекомендация. Для очередной строки при ее просмотре создать массив входящих в нее слов, преобразовать его в соответствии с условием задачи, а затем переписать в выходной файл.

---

37. Входной текст содержит список целых десятичных чисел, разделенных между собой запятыми. Удалить из текста числа, которые входят в диапазон $[m,n]$, $n > m \geq 0$, а другие числа преобразовать в двоичную систему счисления. Значения целочисленных переменных *m* и *n* ввести с клавиатуры.

---

38. В состав входного текста входят русские слова и целые десятичные числа без знака. Удалить из текста те слова, которые начинаются и заканчиваются согласной буквой, а каждое число заменить его цифровым корнем. При этом учитывать, что прописные и строчные буквы в данном тексте эквивалентны.

Примечание. Цифровым корнем называют такую сумму цифр целого числа, которая после некоторого количества сложений изображается одной цифрой. Пример:

$$853977 \to 8+5+3+9+7+7=39 \to 3+9=12 \to 1+2=3$$

---

39. Входной текст содержит список десятичных констант с фиксированной точкой. Константы разделены между собой запятыми. Перенести в выходной файл список неправильных констант, разделив их на следующие группы по типу ошибки:

- использование недопустимых символов;
- наличие в одной константе нескольких разделяющих точек;
- наличие больше одного знака ‘+’ или '-';
- положение знака '+' или '-' в середине или в конце константы.

Каждая группа неправильных констант должна начинаться с новой строки выходного файла.

Примечание 1. Рекомендуется 4 раза выполнить просмотр входного файла, пересылая поочередно в выходной файл константы одной группы.

Примечание 2. Константа может включать в себя несколько ошибок. Тогда одна и та же константа будет записана в несколько групп.

---

40. Слова входного текста записаны прописными и строчными латинскими буквами, которые считаются эквивалентными. Слова разделены между собой одним или несколькими пробелами. Перенести в выходной файл слова, которые при перестановке букв могут стать палиндромами, разделяя их между собой запятой с пробелом.

Примечание. Слово может быть палиндромом, если

- количество каждой буквы в слове четное;
- среди букв слова есть лишь одна, количество которой нечетное.

---

41. Входной текст содержит список слов, записанных прописными и строчными латинскими буквами. Слова разделены между собой запятыми. Перенести в выходной файл слова, длина которых превышает три символа и при этом в составе слова имеется не менее двух гласных букв. В преобразованном слове первая буква должна быть прописной, остальные – строчными. Слова в выходном файле должны разделяться запятой с пробелом.

См.примечание к п.36.

---

42. Входной текст содержит слова, а также десятичные и двоичные числа. Признаком двоичного числа является буква 'B', записанная после его цифр. Слова и числа разделены между собой одним или несколькими пробелами. Заменить двоичные числа на

сумму их цифр, записав эту сумму также в двоичной системе счисления. Удалить из текста десятичные числа, которые имеют меньше трех цифр. Новый текст переслать в выходной файл, разделяя в нем слова и числа запятой с пробелом.

---

43. Входной текст содержит список арифметических констант в форме целых десятичных чисел, разделенных между собой запятыми. Уменьшить значение каждой константы в 10 раз и записать ее в выходной файл. Учесть такие частные случаи:

- в конце константы имеются нули;
- константа не кратна 10 и имеет больше одной цифры;
- константа имеет лишь одну цифру, но не равна нулю;
- константа равна нулю.

Например, для списка ‘550, 234, 7, 0' получим '55, 23.4, 0.7, 0'.

Указание. Для преобразования констант не использовать процедуры `Val` и `Str`.

---

44. Входной текст содержит несколько пар открывающих и закрывающих скобок. Перенести в преобразованный текст фрагменты входного текста, которые содержатся во всех внутренних скобках. Если в этих фрагментах имеются пробелы, то их необходимо удалить. Фрагменты разделять между собой точкой с запятой и пробелом. Например, для входного текста 'AB* (3+5/C8* (4-16*XYZ)+1)*(3-51*S*(6-8 * K18))' получим 4-16*XYZ; 6-8*K18. При этом текст, который надо переносить, не может иметь продолжения на следующей строке.

---

45. Входной текст содержит слова на русском языке и целые десятичные и восьмеричные числа, разделенные между собой одним или несколькими пробелами. Признаком восьмеричного числа является буква 'X', записанная после последней цифры числа. Удалить из текста десятичные числа, которые имеют меньше трех цифр. Восьмеричные числа преобразовать в десятичную систему счисления, после чего цифры полученного числа переставить в обратном порядке. Преобразованные числа не должны содержать незначащих нулей. Слова и числа в выходном файле разделять запятой с пробелом.

Пример. Исходный текст

4875 слово 384X 25 квадрат 41X 202X 148 12X 7 723 111X

после преобразования будет иметь вид

4875, слово, 62, квадрат, 33, 31, 148, 1, 723, 37

---

46. Входной текст содержит слова, которые состоят из строчных латинских букв, целые десятичные числа и разделители (пробел, запятая, точка, точка с запятой). Считая, что длина отдельного слова не превышает 16 символов, сформировать таблицу слов входного текста, указав в ней количество повторений каждого слова. Таблицу сгруппировать по алфавиту слов. После этого перенести в выходной файл лишь слова, которые не повторяются в исходном тексте, разделяя их между собой запятой с пробелом.

---

47. Входной текст содержит список десятичных чисел, в который входят целые числа и числа с фиксированной точкой (целая и дробная части числа разделены точкой). Элементы списка разделяются запятыми. Некоторые числа могут содержать в явном виде знаки '+' или '-'. Определить количество положительных чисел, количество отрицательных чисел, а также частоту каждой цифры в заданном списке чисел (в процентах, в дробной части две цифры). После этого числа, которые имеют в своем составе цифру с наименьшей, но не нулевой частотой частотой повторения, удалить из текста. Числа в выходном файле разделять между собой двумя пробелами.

---

48. Входной текст содержит список десятичных констант с фиксированной точкой, при этом каждая константа имеет явным образом выраженные целую и дробную части (в частном случае дробная часть может быть нулевой). Элементы списка разделены между

собой запятыми. Необходимо в каждой константе удалить незначащие нули в целой и дробной частях, если в ее составе имеются такие нули. Если дробная часть константы нулевая, то удалить также разделяющую точку. Учесть, что в целой части константы всегда должна быть хотя бы одна цифра. В выходном файле после любой розделяющей запятой записать один пробел. Например, для списка констант '00035.4500,0000.40,72.000,0.0' получим '35.45, 0.4, 72, 0'.

---

49. Слова текста содержат лишь прописные и строчные русские буквы. Эти слова разделены между собой одним или несколькими пробелами. Преобразовать каждое слово таким образом, чтобы в каждой паре его букв (1-2, 3-4, 5-6 и т.д.) эти буквы были расположены по алфавиту. Если слово имеет нечетное количество букв, то последнюю букву не анализировать. Например, для слова 'пОслеДоватьНость' получим 'ОплсДевоателНьсоть'. Если первая и последняя буквы преобразованного слова – гласные, то удалить такое слово из текста.

---

50. Во входном тексте встречаются переносы слов. Необходимо преобразовать этот текст таким образом, чтобы любое его слово находилось в одной строке (то есть избавиться от переносов). После устранения знаков переноса выравнять преобразованный текст по правой границе.

Рекомендация.

Целесообразно преобразование текста выполнять по мере его переноса в выходной файл.

Предположим, что после последнего слова очередной строки записан символ переноса «-». Тогда это слово не переносить в выходной файл, а записать в буферную строку-переменную Buf, а после чтения следующей строки текста присоединить его к первому слову этой строки. При этом строке-переменной Buf присвоить пустое значение (это в дальнейшем будет служить признаком того, что в данный момент нет фрагмента слова, который нужно присоединить к первому слову очередной строки).

---

51. Для разделения слов во входном тексте на русском языке используются пробел, точка, запятая, точка с запятой, двоеточие и тире, при этом любой из разделителей может быть окружен слева и (или) справа одним или несколькими пробелами. Для записи слов используются прописные и строчные буквы, которые считаются эквивалентными. Перенести в выходной файл слова, которые начинаются с гласной буквы, причем эта буква в слове больше не повторяется. Слова, которые имеют меньше трех литер, не переносить. В выходном файле каждое слово записывать с прописной буквы, остальные буквы слова должны быть строчными. Слова в выходном файле разделять между собою запятой с пробелом.

---

52. В состав входного файла входят слова, записанные строчными латинскими буквами, и десятичные числа с фиксированной точкой. Как разделители применяются пробел, запятая и точка с запятой. В каждом числе заменить отдельно целую и дробную части суммой цифр, которые входят в состав этих частей. Если при этом будет обнаружено, что целая часть числа по своему значению меньше дробной, то удалить такое число из текста.

---

53. Слова в тексте разделены между собой пробелом, запятой или точкой, причем после запятой и точки всегда записывается один пробел. В начале любого абзаца текста предполагается пять пробелов. Строки текста в общем случае не выравнены по правой границе. Отредактировать текст, выравняв по правой границе каждую его строку, кроме последней в абзаце. При выравнивании добавлять в промежутки между словами необходимое количество пробелов, при этом все промежутки должны отличаться между собой не более чем на один пробел.

Примечание. *k*-ая строка является последней в абзаце, если (*k*+1)-ая строка имеет абзац или же сама *k*-ая строка является последней в тексте.

---

54. Слова входного текста записаны прописными и строчными латинскими буквами, которые считаются эквивалентными. Слова разделены между собой одним или несколькими пробелами. Дополнить каждое слово минимальным количеством букв таким образом, чтобы при перестановке букв оно могло стать палиндромом. Слова в преобразованном тексте разделять между собой запятой с пробелом.

См. примечание к задаче 40.

---

55. В состав входного текста входят слова и целые десятичные числа, разделенные между собой одним или несколькими пробелами. Для записи слов используются прописные и строчные латинские буквы. При преобразовании текста требуется:

- удалить из каждого слова буквы, которые имеют нечетный порядковый номер в латинском алфавите;
- если число имеет нечетное значение, то добавить к нему единицу.

---

56. Входной текст на русском языке состоит из слов, целых десятичных чисел и разделителей (пробел, точка, запятая, точка с запятой и двоеточие, при этом любой из них может быть окружен слева и (или) справа одним или несколькими пробелами). Заменить числа, значение которых меньше 40, русскими словами. Например, вместо 6 записать “шесть”, вместо 14 – “четырнадцать”, вместо 31 – “тридцать один” и т.д.

Указание. В программе должно быть использовано минимально возможное количество строковых констант. Это означает, что не допускается для каждого числа иметь свою строковую константу (‘двадцать один’, ‘двадцать два’ и т.д.). В программе числа должны конструироваться из отдельных элементов (‘двадцать’+‘один’, ‘двадцать’+‘два’ и т.д.).

---

57. Входной текст включает в себя слова, записанные строчными и прописными латинскими буквами, и целые десятичные числа. Как разделитель в тексте используется пробел или группа пробелов. В каждом слове переставить буквы в алфавитном порядке, заменив при этом прописные буквы строчными, а в каждом числе переставить цифры в порядке их увеличения. Если при преобразовании числа получаются незначащие нули, то удалить их из состава числа.

Примечание. В таблице ASCII строчные латинские буквы расположены через 32 позиции после прописных. Следовательно, 'a'=chr(ord('A')+32), 'b'=chr(ord('B')+32) и т.д.

---

58. Входной текст содержит список арифметических констант в форме целых десятичных чисел и в форме десятичных чисел, целая и дробная часть которых разделены точкой. Элементы списка разделены между собой запятыми. Нужно увеличить значение каждой константы в 10 раз, не используя при этом преобразование чисел с помощью процедур `Val` и `Str`. В новом списке после каждой запятой записать один пробел.

Например, для входного списка '52,6.6,6.66,77.777,0,0.005' получим '520, 66, 66.6, 777.77, 0, 0.05'.

---

59. Входной текст содержит список целых десятичных чисел, разделенных между собой запятыми. Преобразовать эти числа в восьмеричную систему счисления, после чего переставить цифры числа в обратном порядке. Если при этом образуются незначащие нули, то удалить их из состава числа. После цифр числа записать литеру 'X' как признак восьмеричного числа. В новом списке после каждой запятой записать один пробел.

---

60. В тексте записаны слова на русском языке (при этом используются как прописные, так и строчные буквы), а также целые числа в десятичной и двоичной системах счисления. Как признак двоичного числа после его последней цифры записывается буква

'B'. Для разделения элементов текста используются пробел, запятая и точка, при этом любой из разделителей может быть окружен одним или несколькими пробелами. Необходимо преобразовать двоичные числа в восьмеричную систему счисления, записав дополнительно как признак числа после его последней цифры букву 'X'. Кроме этого, из состава текста нужно удалить слова, в которых меньше двух гласных букв.

Рекомендация.

Разделить двоичное число справа налево на триады. Если в последней слева триаде меньше трех двоичных цифр, то дополнить ее незначащими нулями. После этого заменить каждую триаду одной восьмеричной цифрой.

---

61. Слова в тексте разделены одним из следующих символов: пробел, точка, запятая, точка с запятой и двоеточие, при этом любой из них может быть окружен слева и (или) справа одним или несколькими пробелами. Для записи слов используются строчные латинские буквы. Определить среднюю длину *S* слов входного текста и перенести в преобразованный текст слова, длина которых превышает значение *S*. В исходном тексте могут быть также десятичные числа. В этом случае в выходной файл переносить только те из них, которые не содержат разделяющей точки. Слова в новом тексте разделять между собой запятой с пробелом.

---

62. В словах входного текста используются только прописные и строчные латинские буквы, а сами слова разделяются между собой одним или несколькими пробелами. Сформировать таблицу двухбуквенных сочетаний, которые начинаются из буквы 'a' ('aa', 'ab', 'ac' и т.д.), указав при этом количество повторений любого из них. Считать, что в данном случае прописные и строчные буквы эквивалентны. Например, в слово 'caArtaj' входят сочетания 'aa', 'ar' и 'aj'. Определить, какое из сочетаний наименее часто повторяется (но не равно нулю), и удалить из текста слова, в состав которых входит такое сочетание.

---

63. Входной текст содержит целые десятичные числа, разделенные между собой одним или несколькими пробелами. Преобразовать текст таким образом, чтобы сначала были записаны все четные, а потом – все нечетные числа, при этом относительная последовательность чисел в каждой из двух групп не должна отличаться от входной. Нулевые числа в выходной файл не переносить. Числа в преобразованном тексте разделять между собой запятой с пробелом.

Рекомендация. Создать буферные файлы F1 и F2; в файл F1 пересылать четные числа, а в F2 – нечетные. После этого в F1 дописать содержимое файла F2, а F2 уничтожить.

---

64. Входной текст содержит список арифметических констант в форме десятичных чисел с плавающей запятой. Элементы списка разделены между собой одним или несколькими пробелами. Перенести в преобразованный текст список констант, которые не превышают по модулю 1. При преобразовании чисел процедуры `Val` и `Str` не использовать. Элементы нового списка разделять между собой запятой с пробелом. Например, в списке 1.2E+12-43.677E-7 3E+01 2567.234E4 0.00087E3 14.6E-5 6.6E+33 такими константами являются -43.677E-7, 0.00087E3, 14.6E-5.

---

65. Входной текст содержит список арифметических констант в форме десятичных чисел с фиксированной точкой. Числа могут иметь знак '+' или '-'. Элементы списка разделены между собой одним или несколькими пробелами. Если дробная часть константы имеет больше трех цифр, то выполнить ее округление до трех цифр, не применяя для этого процедуры `Val` и `Str`. Если в дробной части нет значащих цифр (например, 5.00), то удалить такую константу из текста.

---

66. Для разделения слов во входном тексте на русском языке используются пробел, точка, запятая, точка с запятой, двоеточие и тире, при этом любой из разделителей может

быть окружен слева и (или) справа одним или несколькими пробелами. Для записи слов используются прописные и строчные буквы, считающиеся эквивалентными. Перенести в выходной файл из входного текста слова, которые начинаются и заканчиваются одной и той же буквой, причем эта буква в слове больше не повторяется. Анализу подлежат слова, которые имеют не меньше трех символов.

---

67. Во входном тексте на русском языке встречаются числа без знака, после которых указано слово “грн.”. Заменить каждый такой стоимостной показатель значением в “грн.” и “коп.”, при этом для записи значения в “коп.” всегда использовать две цифры, а после слова “грн.” записывать один пробел. Если исходное число содержит более двух цифр в своей дробной части, то эта часть должна быть округлена до двух цифр.

Примеры:

3,8 грн. = 3 грн. 80 коп.; 0,56 грн. = 0 грн. 56 коп.; 28 грн. = 28 грн. 00 коп.;

5,628 грн = 5 грн 63 коп.; 55,0532 грн = 55 грн 05 коп.

---

68. Текст содержит арифметическое выражение, в состав которого входят идентификаторы простых переменных, идентификаторы переменных с индексами, десятичные константы, знаки арифметических операций (в том числе **`div`** и **`mod`**) и круглые скобки. Перенести в преобразованный текст отдельно список идентификаторов простых переменных и список идентификаторов переменных с индексами, при этом перед каждым списком в отдельной строке по ее середине записать соответствующие заголовки. Элементы выходных списков разделять между собой запятой с пробелом.

Примечание. См. рекомендацию к задаче 63.

---

69. В тексте записаны слова на русском языке, разделенные между собой пробелом, запятой или точкой, причем после запятой и точки может быть записан один пробел. Для записи слов используются как прописные, так и строчные буквы, считающиеся эквивалентными. Проанализировать слова этого текста. Если слово имеет нечетное количество букв, то добавить в его конец такую букву, которая отсутствует в данном слове, но находится как можно ближе к началу алфавита. Например, для слов 'завод', 'барабан','аббревиатура' получим 'заводб', 'барабанв','аббревиатураг'.

---

70. Входной текст содержит слова, которые состоят из строчных латинских букв, и целые десятичные числа. Как разделители в тексте используются пробел или группа пробелов. Перенести в выходной файл слова входного текста в такой последовательности: первое слово; ближайшее следующее слово, первая буква которого совпадает с последней буквой первого слова; ближайшее следующее слово, первая буква которого совпадает с последней буквой записанного до этого слова и т.д. Для оставшихся неиспользованными слов входного текста повторять описанную процедуру до тех пор, пока не будут перенесены в выходной файл все слова входного текста. Слова в преобразованном тексте разделять между собой запятой с пробелом.

Например, для текста

abc ddfgt vbyu 45 ujhf tbt jiikjh -786 yhgrv 385 fghr vbfr -5678 hatyuhg 12343 cgfdsa rsuytre tvbnm error afghrty

получим

abc, cgfdsa, afghrty, ddfgt, tbt, tvbnm, vbyu, ujhf, fghr, riuytre, error, jiikjh, hatyuhg, yhgrv, vbfr

Рекомендация. Обработку текста целесообразно выполнять в такой последовательности:

1) Удалить из текста все числа.
2) Просматривая строки текста, переносить в выходной файл слова входного текста в соответствии с условием задачи. При этом любое из перенесенных слов сразу же удалять из входного текста.

3) Пункт 2 повторять до тех пор, пока все строки входного текста не станут пустыми.

---

71. В тексте на русском языке записаны слова и целые десятичные числа, разделенные между собой одним или несколькими пробелами. Перед числами может быть записан знак '+' или '-'. Удалить из текста слова, которые имеют нечетную сумму порядковых номеров своих букв в алфавите. Удалить также знак '+', если он записан перед числом.

---

72. Во входном тексте задан список целых десятичных чисел, разделенных между собой одним или несколькими пробелами. Для каждого числа указан его знак, записанный после последней цифры числа. Требуется выполнить следующее преобразование чисел:

- знак '-' записать перед первой цифрой числа;
- знак '+' удалить из состава числа;
- переставить цифры числа в обратном порядке, если число имеет не меньше двух цифр;
- если после перестановки цифр появились незначащие нули, то удалить такие нули;
- если число равняется нулю, то оно не должно иметь никакого знака.

Числа в новом тексте разделять между собой запятой с пробелом.

Например, для списка '345+ 65400- 0+ 76000- 0- 12345+ 7+ 8-' получим ‘543, -456, 0, -67, 0, 54321, 7, -8’.

---

73. Слова текста разделены между собой одним или несколькими пробелами. Для записи слов используются прописные и строчные латинские буквы, считающиеся эквивалентными. В каждой строке определить номера и значения слов, которые имеют между собой соответственно максимальное и минимальное расстояния. Если в строке несколько пар таких слов, то указать первую из них. Запись в выходной файл должна иметь такой вид:

Строка (№ входной строки) Max = ... (пара слов и их номера) Min = ... (пара слов и их номера)

Если в строке нет слов одинаковой длины, то в выходной файл записать:

Строка (№ входной строки) Нет слов одинаковой длины

Строки выходного файла не выравнивать по правой границе.

Примечание. Расстояние между словами одинаковой длины – это количество позиций, в которых отличаются эти слова. Если в строке имеются одинаковые слова, то расстояние между ними равно нулю.

---

74. Входной текст содержит слова, записанные на русском языке, и целые десятичные числа. Как разделители в тексте используются пробел, запятая, точка, точка с запятой и двоеточие, при этом любой из них может быть окружен слева и (или) справа одним или несколькими пробелами. Требуется удалить из состава текста слова, которые имеют нечетное количество букв, а для чисел с четной суммой их цифр добавить единицу.

---

75. Входной текст содержит два предложения, слова которых разделены между собой одним или несколькими пробелами. Для записи слов используются прописные и строчные латинские буквы, считающиеся эквивалентными. Каждое предложение заканчивается точкой с пробелом и может занимать несколько строк. В строке, где заканчивается первое предложение, может начинаться второе предложение. Требуется перенести в выходной файл слова, общие для обоих предложений. Для каждой такой пары в строке выходного файла записывать название слова и в скобках через тире порядковые номера этого слова в первом и втором предложении. Если во втором предложении имеется несколько слов, совпадающих со словом первого предложения, то в выходном файле должно быть столько же сообщений о парах слов. При повторении слов в первом

предложении должно учитываться лишь первое из них. Сообщения в выходном файле разделять между собой запятой с пробелом.

Рекомендация.

Считая, что длина слова не превышает 20 символов, при просмотре текста создать для первого и второго предложений отдельные массивы слов, после чего анализировать эти массивы и переносить в выходной файл соответствующие сообщения.

---

76. Входной текст содержит список арифметических констант в форме десятичных чисел с плавающей запятой. Элементы списка разделены между собой одним или несколькими пробелами. Преобразовать данный список, увеличив порядок каждой положительной константы на 1 и уменьшив порядок каждой отрицательной константы также на 1. Нулевые константы не преобразовывать. Элементы выходного списка разделять между собой запятой с пробелом. Например, для списка '5.1E22 0.08E-11 -12.34E-01 -333E0 0.0E0 1E+1' получим '5.1E23, 0.08E-10, -12.34E-02, -333E-1, 0.0E0, 1E+2'.

---

77. Слова текста разделены между собой одним или несколькими пробелами. Для записи слов используются прописные и строчные латинские буквы, считающиеся эквивалентными. В выходной файл перенести отдельно из каждой строки пары слов одинаковой длины, которые отличаются между собой лишь одной буквой (эта буква не обязательно занимает одну и ту же позицию в первом и втором словах). При обнаружении такой пары слов в выходной файл записать через тире эти слова, а после каждого из них в круглых скобках записать номер строки и порядковые номера слов в тексте. Пары слов разделять между собой запятой с пробелом.

Рекомендация.

При чтении очередной строки создать для нее массив слов, после чего анализировать этот массив и переносить в выходной файл соответствующие сообщения.

---

78. Во входном файле записано несколько предложений, каждое из которых заканчивается точкой с пробелом. Предложение может занимать несколько строк входного текста. В той строке, где заканчивается одно предложение, может начинаться следующее предложение. Слова в предложении записываются на русскои языке, при этом первое слово предложения начинается с прописной буквы. Слова разделяются между собой одним пробелом. Требуется в каждом предложении поменять местами первое и последнее слова, изменив соответствующим образом их первые буквы. Если предложение имеет меньше трех слов, удалить его из состава текста.

Рекомендация.

Считая, что длина слова не превышает 20 символов, для очередного предложения формировать массив слов, преобразовать этот массив в соответствии с условием задачи и перенести его в выходной файл.

---

79. Слова входного текста разделяются между собой пробелом, запятой, точкой, точкой с запятой и двоеточием, при этом каждый из разделителей может быть окружен слева и (или) справа одним или несколькими пробелами. Для записи слов используются прописные и строчные буквы латинского алфавита, считающиеся эквивалентными. Определить среднее относительное количество *S* согласных букв в словах текста, после чего удалить из текста слова, в которых относительное количество согласных букв превышает значение *S*.

---

80. Входной текст – это произвольная последовательность прописных и строчных латинских букв и разделителей (пробел, точка, запятая, точка с запятой, двоеточие, тире). Прописные и строчные буквы в данном случае не считаются эквивалентными. Выполнить сжатие текста, заменив каждую последовательность, которая состоит из более чем трех вхождений одного символа, на $k(s)$, где s – повторяемый символ, а $k>3$ – количество его повторений.

Например, для последовательности

'aaaaUUUUUUkktddddddddd.....rrrr,,,,sffjjjjjj yyyyyyyy..…yyy'

получим

‘(4)a(6)Ukkt(8)d(5).(4)r(4),sff(6)j(6) (8)y(6).yyy’.

---

81. В состав входного текста входят слова, записанные на русском языке, и целые десятичные числа без знака. Как разделители здесь используются пробел, точка, запятая, точка с запятой, двоеточие и тире, при этом любой из них может быть окружен слева и (или) справа одним или несколькими пробелами. Перенести в выходной файл слова, которые имеют больше двух подряд расположенных согласных букв, а также числа, которые имеют больше двух подряд одинаковых цифр. Элементы выходного файла должны быть разделены между собой запятой с пробелом.

---

82. Входной текст содержит список десятичных чисел, целая и дробная часть которых разделены точкой. Как разделитель между элементами списка используется запятая. Преобразовать числа входного текста в восьмеричную систему счисления, округляя при необходимости их дробную часть до трех цифр. После этого переставить отдельно цифры целой и дробной частей в обратном порядке.. Если при этом будут обнаружены незначащие нули, то удалить их из состава числа. Если в дробной части числа не останется значащих цифр, то удалить также разделяющую точку. Элементы преобразованного текста разделять между собой запятой с пробелом.

---

83. Текст включает в себя список десятичных чисел, целая и дробная часть которых разделены точкой. Как разделитель элементов списка используется запятая, после которой записан один пробел. После последнего числа списка запятая не ставится. Если в дробной части числа меньше 4-х цифр, то дополнить ее нулями до 4-х цифр; если эта часть имеет больше 4-х цифр, то округлить ее до 4-х цифр. Если в дробной части числа нет значащих цифр, то удалить такое число из списка. В последнем случае необходимо удалить также запятую с пробелом. Если же изъятию подлежит последнее число, то в этом случае нужно удалить запятую с пробелом перед этим числом.

---

84. В состав текста входят слова, разделенные одним или несколькими пробелами. Для записи слов используются прописные и строчные латинские буквы, которые считаются эквивалентными. Определить максимальную $S_{\max}$ характеристику слов и ее среднее значение S_{cp}. После этого удалить из текста слова, которые имеют характеристику $S < S_{cp}$, а после преобразованного текста записать в выходном файле с новой строки список слов, которые имеют $S = S_{\max}$. Элементы последнего списка разделять между собой запятой с пробелом.

Примечание. Характеристика слова – это длина максимальной серии (последовательности одинаковых символов), что содержится в нем.

*Пример.*Для строки

`Rtttaaaab ksssbbbLLL NnnnnNqt HHrtedddddddDDcd`

имеем характеристики

$$S_1 = 4; \quad S_2 = 3; \quad S_3 = 6; \quad S_4 = 9; \quad S_{\max} = 9; \quad S_{cp} = 5{,}5$$

---

85. Входной текст содержит список целых десятичных чисел, разделенных между собой одним или несколькими пробелами. Преобразовать текст таким образом, чтобы сначала были записаны все положительные числа, а потом – все отрицательные, при этом относительная последовательность чисел в любой из групп не должна отличаться от входной. Каждую из двух групп чисел записывать с новой строки выходного файла. Учесть частный случай, когда во входном списке нет положительных или отрицательных чисел. Нулевые элементы в новый список не включать. После преобразованного текста с новой

строки записать список одинаковых по модулю чисел, сравнивая для этого элементы полученных двух групп чисел. В новый список каждое число должно быть записано лишь один раз. Элементы этого списка разделять между собой запятой с пробелом.

Примечание. См. рекомендацию к задаче 63.

---

86. Входной текст содержит слова и целые десятичные числа, разделенные между собой одним или несколькими пробелами. Для записи слов используются прописные и строчные латинские буквы, которые считаются эквивалентными. Анализируя отдельно каждую строку, перенести в выходной файл те слова, которые больше не повторяются в данной строке. Список слов должен быть сплошным, то есть не допускается записывать слова, содержащиеся в очередной входной строке, с новой строки выходного файла. Элементы этого списка разделять между собой запятой с пробелом.

Рекомендация. При сравнении слов данной строки повторяемое слово изымать из строки, чтобы оно не было потом воспринято как неповторяемое.

---

87. Входной текст содержит слова, которые состоят из латинских букв, целые десятичные числа и разделители (два пробела или запятая с пробелом). Если в слове есть изолированные подслова вида *yaz,* то заменить их на подслова вида *zay,* где *y,z* – согласные буквы, *a* – гласная буква. Изолированность в данном случае означает, что для двух смежных триад вида *yaz* буква *z* не является общей. Если в слове нет подслов *yaz,* которые подлежат инвертированию, то такое слово нужно удалить из текста.

Пример. Для последовательности слов 'rust, sdfehgadi, trwd, kosuhpt' получим 'surt, sdhefdagi' (два последних слова будут удалены).

---

88. Текст содержит сплошную последовательность цифр. Каждые две смежные цифры a_i и a_{i+1} (i = 1, 3, 5, …) определяют номер буквы латинского алфавита (при этом значение $a_i, a_{i+1} = 00$ означает пробел, значение $a_i, a_{i+1} > 26$ ничего не означает, его нужно игнорировать). Если в конце строки оказалась неполная пара (одна цифра), то дописать перед этой цифрой нуль.

Каждую строку входного файла после ее преобразования размещать в отдельной строке выходного файла. Выравнивание строк выходного файла по правой границе не производить.

---

89. Входной текст содержит слова и десятичные числа, целая и дробная часть которых разделены точкой. Числа могут иметь знак '+' или '-'. Для записи слов используются буквы русского алфавита. Элементы текста разделены между собой одним или несколькими пробелами. Если целая часть числа имеет больше 4-х цифр, то переставить цифры этой части в обратном порядке, удалив незначащие нули, если они появились при такой перестановке. Удалить также из текста слова, которые имеют меньше трех букв.

---

90. Элементами входного текста являются слова и целые десятичные числа, разделенные между собой одним или несколькими пробелами. Для записи слов используются строчные и прописные латинские буквы, которые считаются эквивалентными. Анализируя отдельно каждую строку, удалить из ее состава пары слов, из которых одно является инверсией второго (например, 'aBcdEf' и 'Fedcb'). Если в этой же строке имеется пара чисел с таким же свойством, то заменить первое из них их суммой, а второе удалить из строки (например, в паре '1234' и '4321' первое число примет значение '5555', а второе будет удалено из строки; в паре ‘888’ и ‘888’ первое число будет заменено значением ‘1776’).

---

91. Входной текст содержит слова, которые состоят из строчных и прописных латинских букв, и целые десятичные числа, которые могут иметь знак '+' или '-'. Разделителем в тексте является пробел или группа пробелов. Строчные и прописные буквы

считаются эквивалентными. Удалить из текста слова, которые образованы не более чем двумя буквами (например, 'ab', 'abBAba', 'ssSs') и числа, которые имеют длину больше 4-х символов и при этом все цифры в числе разные (например, '12345', '-852361').

---

92. В состав входного текста входят слова, записанные прописными и строчными русскими буквами и разделенные между собой одним или несколькими пробелами. В каждой строке переставить в обратном порядке слова, а в словах – их буквы, одновременно удалив из него те слова, которые созданы лишь одной буквой (например, 'a', 'aaAAa', ‘ссСссС’ и т.п.).

---

93. Текст содержит список десятичных чисел в виде мантиссы и порядка. Числа разделены между собой запятой с пробелом. Часть этих чисел имеет знаки '+' или '-' перед мантиссой и (или) порядком. Требуется для положительных мантиссы и порядка поставить в явном виде знак '+', если такой знак отсутствует. Удалить из списка отрицательные числа, которые по модулю имеют значение меньше 10, не используя при этом процедуры `Val` и `Str`. Например, для списка '+1.12E05, -3.34E0, 543.35E+1, -3.34E02, +6.6E+32, 6.6E03, -1.1E-08, -1.1E+4' получим '+1.12E+05, +543.35E+1, -3.34E+02, +6.6E+32, +6.6E+03, -1.1E+4'.

---

94. В тексте записаны слова, состоящие из прописных и строчных латинских букв, которые считаются эквивалентными, и целые десятичные числа. Как разделители здесь используются пробел, точка, запятая, точка с запятой и двоеточие, при этом каждый из них может быть окружен слева и (или) справа одним или несколькими пробелами. В каждом из слов удалить сначала символы-серии, если они имеются, а затем символы, которые уже встречались в этом слове. В конце каждого числа дописать наименьшую из возможных цифр, которые отсутствуют в этом числе. Например, для чисел ’35, 1230456789, 610’ получим ‘350, 1230456789, 6102’, а для слов ‘abcuucxzzzuytytytabrcc, abcde, abcdebcd’ - ‘abcxuytr, abcde, abcde’.

Примечание. Серией считать последовательность одинаковых символов.

---

95. В тексте записаны слова, состоящие из прописных и строчных латинских букв, которые считаются эквивалентными, и целые десятичные числа. Как разделитель здесь используются пробел или группа пробелов. Если в слове больше двух букв и их количество четное, то удалить из него две разные, но ближайшие к началу алфавита буквы. Слово, в котором все буквы одинаковы, оставить без изменения. Одновременно в каждом числе удалить те цифры, которые повторяются, кроме первой такой цифры. Например, для текста 'asXUrt qWEry ddDDdd 764565 456789 30000' получим 'sXUt qWEry ddDDdd 7645 456789 30'.

---

96. В тексте записаны слова, которые состоят из прописных и строчных латинских букв, считающихся эквивалентными, и целые десятичные числа. Как разделители здесь используются пробел, точка, запятая, точка с запятой и двоеточие, при этом каждый из них может быть окружен слева и (или) справа одним или несколькими пробелами. Переписать в выходной файл слова, в которых нет подряд расположенных одинаковых букв, а также числа, в которых хотя бы одна пара смежных символов состоит из одинаковых цифр. Элементы нового текста разделить между собой запятой с пробелом. В новом тексте не должно быть двух одинаковых элементов.

Например, для входного текста 'as tTyu, 455; df df 455 srRss 1234 as 455 yutre yyuu ' получим 'as, 455, df, 1234, yutre'.

---

97. В тексте записаны слова, которые состоят из прописных и строчных латинских букв, считающихся эквивалентными, и целые десятичные числа. Как разделители здесь используются пробел, точка, запятая, точка с запятой и двоеточие, при этом каждый из них может быть окружен слева и (или) справа одним или несколькими пробелами. В каждом

слове переставить его буквы по алфавиту. Если в числе больше двух цифр, то переставить их в порядке уменьшения, в противном случае такое число удалить из состава текста.

---

98. Входной текст содержит список целых десятичных чисел, разделенных между собой одним или несколькими пробелами. Преобразовать текст таким образом, чтобы сначала были записаны все отрицательные нечетные числа, а потом – все остальные, кроме нулевых. При этом относительная последовательность чисел в любой из двух групп не должна отличаться от входной.

Примечание. См. рекомендацию к задаче 63.

---

99. В тексте записаны слова, которые состоят из прописных и строчных латинских букв, считающихся эквивалентными, и целые десятичные числа. Как разделитель здесь используется пробел или группа пробелов. Перенести в выходной файл слова, в которых нет удвоения букв, а также числа, в которых имеет место удвоение цифр. Элементы выходного файла разделять между собой запятой с пробелом.

---

100. Во входном тексте содержатся слова и целые десятичные числа, разделенные одним или несколькими пробелами. Для записи слов используются строчные и прописные латинские буквы, в их состав могут входить также цифры. Если в начале слова стоят цифры, то переставить их циклически в конец слова так, чтобы каждое слово начиналось с буквы. Первая буква любого слова, которое остается в тексте, должна быть прописной, остальные – строчными. Например, для слова '367abc8Fg' получим 'Abcd8fg367'. Если в слове нет ни одной цифры, то такое слово удалить из состава текста. Удалить из текста также числа, значение которых меньше 11. Если число больше 10 и имеет нечетное значение, то отнять от него единицу.

См. примечание к задаче 57.