Лабораторная работа № 7

Обработка строк с использованием множественного типа данных

Задание: составить программу заданной обработки массива слов. В процессе обработки использовать множественных тип данных. Заполнение исходных данных – с клавиатуры. Исходный и обработанный массив выводить на экран.

Теоретический материал

Строковый тип данных занимает промежуточное положение между простыми и структурированными типами данных. С одной стороны, данные строкового типа имеют структуру (строка, по сути, – это последовательность символов). С другой стороны, строковый тип, как все простые типы, является стандартным, а ни один структурированный тип не является стандартным. Кроме того, над строками можно выполнять некоторые действия, которые допустимы для данных простых типов и недопустимы для структурированных типов данных. Например, строку `s` можно ввести с клавиатуры или вывести на экран с помощью стандартных процедур ввода-вывода `read(s)` и `write(s)`. В то же время, если описать переменную `s` как массив символов, то для ввода (или вывода) необходимо организовывать цикл, в котором стандартные процедуры ввода-вывода будут применяться к элементу массива `s`, т. е. `read(s[i])` или `write(s[i])`.

Значениями строковых переменных могут быть последовательности различной длины (от нуля до 255 символов, длине 0 соответствует пустая строка).

Для описания данных строкового типа используется зарезервированное слово `string`. По аналогии с любым стандартным типом, это слово можно использовать для описания переменных в разделе переменных следующим образом:

```
var s:string;
```

Такое описание будет означать, что для переменной `s` в памяти зарезервировано место под 255 символов (т. е. максимальная длина строки). Если же максимально возможное значение строковой переменной меньше (например, имя человека обычно содержит не более 20 символов), то можно ограничить длину строки, сэкономив тем самым оперативную память:

```
var name:string[20];
```

Вместе с тем, по аналогии со структурированным типом, можно определить строковый тип в разделе описания типов:

```
type str_20=string[20];
```

Затем использовать данный тип при описании переменных:

```
var name:str_20;
```

Стандартные процедуры и функции для работы со строками приведены ниже.

Функции

`concat`	`объединяет несколько строк`
`copy`	`возвращает подстроку, содержащуюся в строке`
`lenght`	`возвращает длину строки`
`pos`	`осуществляет поиск подстроки в строке`
`ord`	`возвращает код символа`
`chr`	`возвращает символ по его коду`

Процедуры

`delete`	`удаляет подстроку из строки`
`insert`	`вставляет подстроку в строку`
`str`	`преобразует числовое значение в строку цифр`
`val`	`преобразует строку цифр в числовое значение`

Еще один тип данных, который потребуется при выполнении лабораторной работы, – множество. Множество – это структурированный тип данных, представляющий набор взаимосвязанных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое.

Для задания типа-множества следует использовать зарезервированные слова `set` и `of`, а затем указать элементы этого множества, как правило в виде перечисления или диапазона:

```
Type <имя типа> = set of <el1, el2, ..., eln>;
```

Введя тип-множество, можно задать переменные или типизированные константы этого типа-множества. Так же, как и для других структурированных типов, тип-множество можно ввести непосредственно при задании переменных или типизированных констант.

```
var alf:set of char;
```

Множеству можно в программе присвоить значение, которое обычно задается с помощью конструктора множества:

```
alf:=['A'..'Z'];
```

В каждое множество включается и так называемое пустое множество – [ ], не содержащее никаких элементов.

Операции над множествами:

`+` объединение множеств;
`-` разность множеств;
`*` пересечение множеств;
`=` проверка эквивалентности двух множеств;
`<>` проверка неэквивалентности двух множеств;
`<=` проверка того, является ли левое множество подмножеством правого множества;
`>=` проверка того, является ли правое множество подмножеством левого множества;
`in` оператор вхождения, т. е. проверка того, входит ли элемент, указанный слева, в множество, указанное справа.

Результатом операций объединения, разности или пересечения является соответствующее множество, остальные операции дают результат логического типа.

Пример выполнения лабораторной работы

Для примера рассмотрим следующую задачу. Дан массив из `n` слов произвольной длины (длина слова не превышает 80 символов). Символами могут быть буквы латинского алфавита и цифры. Определить сумму цифр во введенном с клавиатуры тексте.

Обобщенный алгоритм решения данной задачи представлен на рис. 7.1.

Программу, как обычно, начнем с заголовка, подключения модуля `crt` (в тексте программы будет использована процедура очистки экрана `clrscr`), разделов описания типов и переменных. Тип `T_str` определяем для работы с массивом слов, тип `T_set` – тип-множество символов.

```
program lab_7;
uses crt;
type T_str=array [1..10] of string[80];
     T_set=set of char;
```

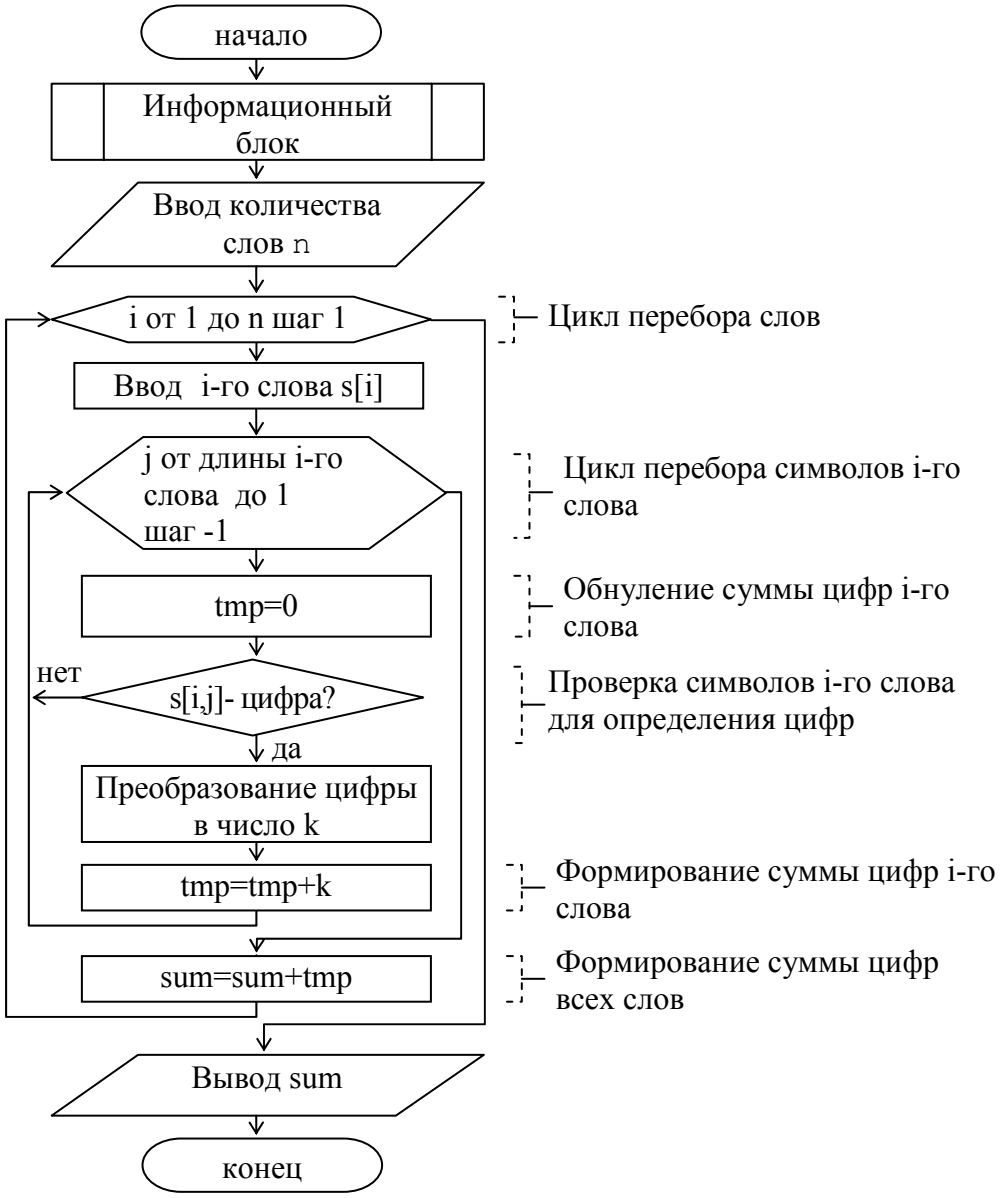

Рис. 7.1. Блок-схема алгоритма определения суммы цифр, содержащихся в массиве слов

В разделе описания переменных объявляются переменные для хранения массива слов, множества символов, которому в дальнейшем присвоим значение допустимых символов, индексов, длины слова и количества слов (описываются целочисленным типом `byte`). Кроме того, в программе понадобятся целочисленные переменные, которые будут использоваться для преобразования символов цифр в число и вычисления необходимых сумм.

```
var s:T_str;      {массив слов}
    alf:T_set;    {множество символов}
    n,            {количество слов}
    i,j:byte;     {индексы}
    cod,          {код ошибки преобразования строки в число}
    k,            {число, полученное преобразованием строки}
```

```
tmp,          {сумма цифр слова}
sum:integer;  {сумма цифр всех слов}
```

Далее следует тело программы. Информационный блок подробно рассматриваться не будет. Перед началом ввода массива слов необходимо присвоить значение переменной множественного типа `alf`. По условию в массиве слов допускаются только латинские буквы и цифры, поэтому с помощью конструктора множеств задаем соответствующее значение. Латинские буквы ограничим лишь диапазоном заглавных букв, поскольку в дальнейшем при проверке символов, входящих в массив слов будет применять функцию преобразования символа в верхний регистр `UpCase`.

```
alf:=['A'..'Z','0'..'9'];
```

Если программист не использует эту функцию, тогда необходимо указывать диапазон как заглавных, так и строчных латинских букв:

```
alf:=['A'..'Z','a'..'z','0'..'9'];
```

Далее необходимо задать количество слов в массиве. Напомним, что при описании типа `T_str` было определено максимально возможное количество слов в массиве, но не указано количество слов, которое будет обрабатываться. Для хранения количества слов в массиве определена переменная n, значение которой можно задать непосредственно в тексте программы, а можно запросить у пользователя. Во втором случае для предотвращения ошибок ввода необходимо предусмотреть проверку корректности вводимых значений:

```
repeat
  write ('Введите количество слов (не более 10) ');
  readln(n);                    {ввод числа слов}
  if not (n in [1..10]) then
  begin
     writeln;
     writeln (’Ошибка ввода!’);
     writeln;
  end;
until (n in [1..10]);
```

Цикл `Repeat` позволяет выполнить операторы тела цикла, по крайней мере, один раз. После ввода клавиатуры количества слов проверяется условие «значение переменной `n` не входит в множество целых чисел от 1 до 10». Если это условие выполняется, то выдается сообщение об ошибке и тело цикла повторяется еще раз, т.к. условие выхода из цикла («значение переменной `n` входит в множество целых чисел от 1 до 10») ложно. Если же пользователь вводит допустимое количество слов, то цикл завершается.

Теперь можно организовывать цикл по вводу и обработке массива слов. Используем для этого цикл `for`. В теле цикла, прежде всего, осуществляется ввод слова с клавиатуры и обнуление переменной, в которой накапливается сумма цифр i-го слова.

```
for i:=1 to n do
begin
   write ('введите ',i,'-е слово ');
   readln(s[i]);        {ввод i-го слова}
   tmp:=0;  {обнуление суммы цифр i-го слова}
```

Далее в блок-схеме алгоритма следует проверка символа на принадлежность его к цифрам, для чего организуется еще один цикл, в котором будет осуществляться перебор символов `i`-го слова. В программе мы несколько усложним алгоритм за счет добавления проверки символов на допустимость значений, т. е. условия, которое будет проверять принадлежность символа к множеству допустимых символов `alf`. В этом используется функция преобразования символа в верхний регистр `UpCase` (строчные буквы становятся заглавными, заглавные – остаются без изменений).

```
      for j:=length(s[i]) downto 1 do
         if UpCase(s[i,j]) in alf then  {проверка допустимости
j-го символа i-го слова}
         begin
```

Если проверяемый символ допустимый, то проверяется его принадлежность к множеству цифр.

```
           if s[i,j] in ['0'..'9'] {если j-й символ i-го слова -
цифра}
           then
```

Когда символ оказывается цифрой, то происходит его преобразование в число и увеличение суммы цифр текущего слова.

```
           begin
               val(s[i,j],k,cod); {преобразование символа в число}
                tmp:=tmp+k; {накапливается сумма цифр i-го слова}
           end
         end
```

Если же проверяемый символ оказался недопустимый, то выдается сообщение об ошибке, обнуляется сумма цифр слова и уменьшается номер слова. Цикл `For` автоматически увеличивает параметр цикла, и такое «топтание» на месте позволит еще раз повторить ввод и обработку ошибочного слова. Команда `break` позволит досрочно выйти из цикла проверки символов і-го слова.

```
         else
         begin
              writeln('Встретился недопустимый символ!');
              writeln;
              tmp:=0;
              dec(i);  {Уменьшение номера слова}
              break;   {Досрочный выход из цикла по j}
         end;
```

После выхода из цикла проверки символов i-го слова происходит увеличение суммы цифр текущего слова.

```
        sum:=sum+tmp;   {накапливается сумма цифр слов}
     end;
```

Когда все слова массива будут обработаны, остается вывести результаты на экран и завершить программу.

```
     writeln('Сумма цифр = ',sum);
     readln;
end.
```

Итак, соединив фрагменты в единое целое, получим текст программы.

```
{Дан  массив слов произвольной длины, но не превышающих 80 симво-
лов.
Символами могут быть буквы латинского алфавита и цифры.
Определить сумму цифр во введенном с клавиатуры тексте.}
program lab_7;
uses crt;
type T_str=array [1..10] of string[80];
     T_set=set of char;
var s:T_str;     {массив слов}
    alf:T_set;   {множество символов}
    n,           {количество слов}
    i,j:byte;    {индексы}
    cod,         {код ошибки преобразования строки в число}
    k,           {число, полученное преобразованием строки}
    tmp,         {сумма цифр слова}
    sum:integer;  {сумма цифр всех слов}
begin
     clrscr;
     writeln;
     writeln('       Автор - Иванов И.П., студент гр. ИСЭд-11');
     writeln('       Вариант № 100');
     writeln('Дан массив слов произвольной длины, но не превышаю-
щих 80 символов. ');
     writeln('Символами могут быть буквы латинского алфавита и
цифры.');
     writeln('Определить сумму цифр во введенном с клавиатуры тек-
сте.');
     writeln;
     alf:=['A'..'Z','0'..'9'];    {задается множество допустимых
значений символов}
     repeat
       write ('Введите количество слов (не более 10) ');
       readln(n);                   {ввод числа слов}
```

```
      if not (n in [1..10]) then
      begin
         writeln;
         writeln (’Ошибка ввода!’);
         writeln;
      end;
    until (n in [1..10]);
    for i:=1 to n do
    begin
       write ('введите ',i,'-е слово ');
       readln(s[i]);        {ввод i-го слова}
       tmp:=0;  {обнуление суммы цифр i-го слова}
       for j:=length(s[i]) downto 1 do
          if UpCase(s[i,j]) in alf then {проверка допустимости j-
го символа i-го слова}
          begin
            if s[i,j] in ['0'..'9'] then {если j-й символ i-го
слова - цифра}
            begin
                val(s[i,j],k,cod); {преобразование символа в число}
                 tmp:=tmp+k;          {накапливается сумма цифр i-
го слова}
            end
          end
          else
          begin
               writeln('Встретился недопустимый символ!');
               writeln;
               tmp:=0;
               dec(i);  {Уменьшение номера слова}
               break;   {Досрочный выход из цикла по j}
          end;
       sum:=sum+tmp;   {накапливается сумма цифр всех слов}
    end;
    writeln('Сумма цифр = ',sum);
    readln;
end.
```

Варианты заданий

1. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Символами могут быть только заглавные латинские буквы. Найти и вывести все гласные буквы (без повторений), которые встретились в двух самых длинных словах.

2. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Символами могут быть только заглавные латинские буквы. Найти и вывести все глухие согласные буквы (без повторений), которые встретились в двух самых коротких словах.
3. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые графические символы. Удалить из массива все слова, содержащие не меньше трех четных цифр.
4. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Символами могут быть только заглавные латинские буквы. Найти и вывести все слова, у которых число гласных букв превышает число согласных
5. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). В качестве символов могут использоваться только арабские цифры. Вывести слово, содержащее наибольшее число нечетных цифр.
6. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). В качестве символов могут использоваться только арабские цифры. Удалить из массива слово, содержащее наибольшее количество четных цифр.
7. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). В качестве символов могут использоваться только арабские цифры. Удалить из массива слово, содержащее наименьшее количество четных цифр.
8. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). В качестве символов могут использоваться только арабские цифры. Вывести те слова, у которых число четных цифр превышает число нечетных.
9. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Символами могут быть только заглавные латинские буквы. Найти слова, содержащие соответственно наибольшее и наименьшее количество гласных букв, и поменять их местами.
10. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Символами могут быть только заглавные латинские буквы. В самом коротком слове все согласные буквы заменить на букву «А».
11. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые графические символы. В слове наименьшей длины удалить все гласные буквы и подсчитать их количество в этом слове.
12. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые графические символы. Все

цифры, содержащиеся в самом длинном слове этого массива, заменить на символ «*».

13. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые графические символы. Вывести все четные цифры, содержащиеся в слове наибольшей длины, и вывести число повторений каждой этой цифры в этом слове.
14. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые графические символы. Найти и вывести слово, содержащее наибольшее количество цифр.
15. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). В слове, котором обнаружено наибольшее количество шипящих букв, заменить их на символ «&».
16. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Вывести все гласные буквы, содержащиеся в слове наибольшей длины, и вывести число повторений каждой этой буквы в этом слове.
17. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Вывести все согласные буквы, содержащиеся в слове наибольшей длины, и вывести число повторений каждой этой буквы.
18. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые графические символы. Подсчитать количество символов во всех словах массива, отличных от заглавных латинских букв.
19. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Слова могут содержать любые символы языка. Найти в массиве и вывести слово, содержащее наибольшее количество слов, отличных от заглавных латинских букв.
20. Дан массив из n слов произвольной длины, но не превышающей 80 символов. Слова могут содержать любые символы языка. Найти и вывести в самом длинном слове массива все символы, отличные от заглавных латинских букв.
21. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Слова могут содержать любые символы языка. Найти слова, содержащие соответственно наибольшее и наименьшее количество цифр, и поменять их местами.
22. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Слова могут содержать любые символы языка. Найти слова, содержащие соответственно наибольшее и наименьшее количество нечетных цифр, и поменять их местами.
23. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Слова могут содержать любые символы языка. Найти слова,

содержащие соответственно наибольшее и наименьшее количество нечетных цифр, и поменять их местами.

24. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Слова могут содержать латинские буквы и цифры. Найти слова, содержащие соответственно наибольшее и наименьшее количество заглавных латинских букв, и поменять их местами.

25. Дан массив из n слов произвольной длины (длина слова не превышает 80 символов). Элементами слов могут быть любые символы. В слове наименьшей длины удалить все четные цифры и подсчитать их количество в этом слове.