7 Программу из примера 8.8 записали следующим образом:

```
var s, p, t: string;
  k, i: integer;
begin
 writeln('Строка s');
 readln(s);
 writeln('Подстрока p');
 readln(p);
 k := 0; i := 1;
 while PosEx(p, s, i) <> 0 do
 begin
  i := PosEx(p, s, i) + 1; k := k + 1;
 end;
 writeln('Встречается', k, 'раз(-а)');
end.
```

Сравните эту программу и программу из примера 8.8, определив, сколько раз выполнится команда цикла в каждой из программ для перечисленных случаев.

1. Строка s := «Не слово хозяин хозяину, а хозяин слову хозяин», подстрока p := «хозяин».
2. Строка s := «Не слово хозяин хозяину, а хозяин слову хозяин», подстрока p := «хозяйка».

*Можно ли подобрать тест, для которого количество выполнений цикла будет одинаковым для обеих программ? Если да, то какой?

8 Получите из слова «ТЕСТИРОВАНИЕ» указанные слова. Для этого используйте команды: `copy`, `delete`, `insert` и операцию сложения строк.

1. РОСТ.
2. НИВА.
3. ТОВАР.
4. ТОСТЕР.
5. ОТВЕРСТИЕ.
6. Придумайте свое слово.

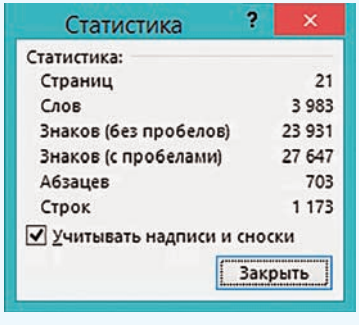
§ 9. Составление алгоритмов обработки строковых величин

9.1. Анализ текста на наличие различных символов

Современные текстовые редакторы позволяют получить статистику по символам и словам в документе (пример 9.1).

Грамотный набор текста предполагает наличие только одного пробела между словами. В правильно набранном тексте количество слов будет на единицу больше, чем количество пробелов.

Пример 9.1. Статистика в документе Word (соответствующая команда на вкладке **Рецензирование**).



Пример 9.2.

V. Программа:

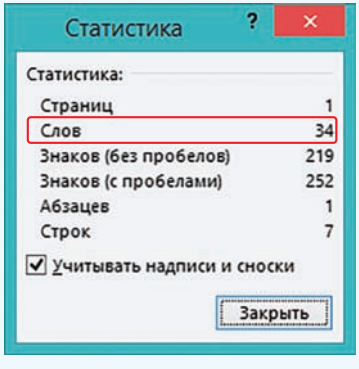
```
var st: string;
    n, k: integer;
begin
  writeln('Введите текст');
  readln(st);
  n := length(st);
  k := 1;
  for var i := 1 to n do
  begin
    if st[i] = ' ' then
      k := k + 1;
  end;
  writeln('В тексте ', k, '
    слов(-о/-а)');
end.
```

VI. Тестирование (для проверки можно скопировать текст абзаца из документа Word).

Окно вывода

```
Введите текст - st
Современные текстовые редакторы позволяют
получить статистику по символам и словам в
документе (пример 9.1). Грамотный набор текста
предполагает наличие одного пробела между
словами. В таком тексте количество слов будет
на 1 больше количества пробелов.
В тексте 34 слов(-о/-а)
```

VII. Анализ результата. Если посмотреть статистику Word для этого абзаца, получим:



Пример 9.2. Написать программу, которая определит количество слов в тексте, если между любыми двумя словами ровно один пробел. Предполагается, что в тексте есть хотя бы одно слово.

Этапы выполнения задания

I. Исходные данные: строка текста `st`.

II. Результат: количество слов `k`.

III. Алгоритм решения задачи.

1. Вводим исходные данные.
2. Определяем длину строки.
3. Задаем начальное значение счетчика `k = 1` (в тексте есть хотя бы одно слово).
4. С помощью цикла **for** просматриваем каждый символ в строке. Если текущий символ — пробел, то увеличиваем значение счетчика количества слов.
5. Выводим результат.

IV. Описание переменных: `st` – `string`, `n`, `k` – `integer`.

Пример 9.3. Написать программу, которая определит, каких знаков препинания в тексте больше — точек или запятых.

Этапы выполнения задания

I. Исходные данные: переменная `st` (текст).

II. Результат: сообщение о том, каких знаков больше.

III. Алгоритм решения задачи.

1. Вводим исходные данные.
2. Определяем длину строки.
3. Инициализируем два счетчика нулями (для точек и запятых).
4. В цикле **for** проверяем каждый символ строки `st`.

4.1. Если встретилась точка, то увеличиваем значение счетчика k1 на 1.

4.2. Если встретилась запятая, то увеличиваем значение счетчика k2 на 1.

5. Сравниваем полученные значения счетчиков и выводим результат.

IV. Описание переменных: st – string, n, k1, k2 – integer.

Для лучшего запоминания маленькими детьми гласных и согласных букв их часто окрашивают в разные цвета: гласные — красным, а согласные — синим (пример 9.4).

Пример 9.5. Написать программу, которая выведет в заданном слове согласные буквы синим цветом, а гласные — красным (в слове не встречаются «ь» и «ъ»). Посчитать количество гласных букв во введенном слове.

Этапы выполнения задания

I. Исходные данные: переменная s (слово).

II. Результат: слово, в котором буквы выводятся разными цветами, и сообщение о количестве гласных букв.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Определяем длину строки.

3. Создаем дополнительную строку, в которой хранятся все гласные буквы: g := 'ЁУЕЭОАЫЯИЮёуеэоаыяию'.

4. Инициализируем счетчик гласных букв k := 0.

5. В цикле **for** проверяем каждый символ слова s.

5.1. Находим позицию текущего символа из строки s

Пример 9.3.

V. Программа:

```
var St: string;
  n, k1, k2: integer;
begin
  writeln('Введите текст');
  readln(St);
  n := length(St);
  k1 := 0; k2 := 0;
  for var i := 1 to n do
  begin
    if St[i] = '.' then
      k1 := k1 + 1;
    if St[i] = ',' then
      k2 := k2 + 1;
  end;
  if k1 > k2 then
    writeln('Точек больше')
  else
    if k2 > k1 then
      writeln('Запятых больше')
    else
      writeln('Количество запятых
          равно количеству точек');
end.
```

VI. Тестирование (для проверки можно скопировать текст абзаца из документа Word).

Окно вывода

```
Введите текст
Для книжных стилей и письменной речи характерны сложные предложения, которые позволяют сделать речь более информативной и выразительной. Если текст состоит в основном из простых предложений, то в нем будут преобладать точки. В сложных предложениях, которые содержат несколько грамматических основ, встречаются запятые.
Запятых больше
```

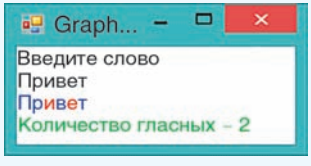
Пример 9.4. Русский алфавит.

Пример 9.5.

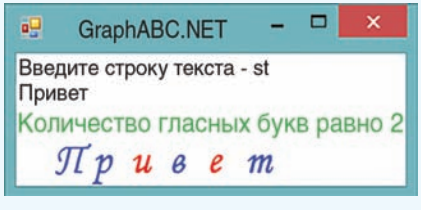
V. Программа:

```
uses GraphABC;
var s, g: string;
   n, k, p: integer;
begin
  writeln('Введите слово');
  readln(s); writeln(s);
  n := length(s);
  g := 'ЁУЕЭОАЫЯИЮёуеэоаыяию';
  for var i := 1 to n do
  begin
    p := pos(s[i], g);
    if p <> 0 then
    begin
      SetFontColor(clRed);
      k := k + 1;
    end
    else
      SetFontColor(clBlue);
    write(s[i]);
  end;
  writeln;
  SetFontColor(clGreen);
  writeln('Количество
гласных - ', k);
end.
```

VI. Тестирование. Ввести слово «Привет». Результат:



Поскольку текст выводится в графическом окне, то можно задавать не только цвет символов, но также шрифт и размер символов. Использование команды `TextOut` позволит выводить символы текста в произвольном месте графического окна. Например, так:



в строке `g` (переменная `p`). Если значение $p \neq 0$, то текущий символ встретился в строке с гласными буквами и, следовательно, сам является гласной. Если $p = 0$, то символ — согласная.

5.2. Если символ является гласной буквой, то выводим его на экран красным цветом и увеличиваем значение счетчика гласных букв на 1, если символ является согласной буквой, то выводим его синим цветом.

6. Выводим результат.

IV. Описание переменных: `s`, `g` – `string`, `n`, `k`, `p` – `integer`.

9.2. Преобразование строк

При записи вещественных чисел на уроках математики в качестве разделителя целой и дробной части используется символ «запятая». В языке программирования Pascal разделителем является точка.

Пример 9.6. Написать программу, которая заменит в строке с вещественными числами запятые на точки. Например, из числа 23,5 должно получиться число 23.5.

Этапы выполнения задания

I. Исходные данные: переменная `st` (введенная строка).

II. Результат: преобразованная строка.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Вычисляем длину строки.

3. В цикле **`for`** проверяем каждый символ строки `st`. Если текущий символ текста запятая, то за-

меняем его на точку. Другие символы оставляем без изменения.

4. Выводим результат.

IV. Описание переменных: `st` – `string`, `n` – `integer`.

Во всех текстовых редакторах реализована функция «заменить». При выполнении этой команды некоторые символы из строки удаляются, а вместо них вставляются другие символы.

Пример 9.7. Написать программу, которая заменит в тексте каждую цифру 2 словом «два».

Этапы выполнения задания

I. Исходные данные: переменная *st* (введенный текст).

II. Результат: преобразованный текст.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Поскольку удалять мы будем один символ, а вставлять три, то длина строки будет изменяться при обработке строки. Количество повторений цикла заранее не известно, поэтому будем использовать цикл **`while`**. Счетчик цикла будет изменяться от 1 до длины строки.

3. В цикле **`while`** проверяем каждый символ текста `st`.

3.1. Если текущий символ текста «2», то удаляем его и вставляем подстроку «два».

3.2. Переходим к следующему символу.

4. Вывод результата.

IV. Описание переменных: `st` – `string`, `i` – `integer`.

При правильном наборе компьютерного текста между любыми двумя

Пример 9.6.

V. Программа:

```
var st: string; n: integer;
begin
 writeln('Введите строку');
 readln(st);
 n := length(st);
 for var i := 1 to n do
  if (st[i] = ',') then
   st[i] := '.';
writeln('Преобразованная строка: ');
 writeln(st);
end.
```

VI. Тестирование.

Окно вывода

```
Введите строку
23,5 34,71  89,234
Преобразованная строка:
23.5 34.71  89.234
```

Пример 9.7.

V. Программа:

```
var st: string;
   i: integer;
begin
 writeln('Введите строку');
 readln(st);
 i := 1;
 while i <= length(st) do
 begin
  if (st[i] = '2') then
  begin
   //замена символов
   delete(st, i, 1);
   insert('два', st, i);
  end;
  //переход к следующему символу
  i := i + 1;
 end;
 writeln('Преобразованная строка: ');
 writeln(st);
end.
```

VI. Тестирование.

Окно вывода

```
Введите строку
В комнате стояли 2 стула и 2 стола.
Преобразованная строка:
В комнате стояли два стула и два стола.
```

Пример 9.8. Выделение лишних пробелов в Word.

При правильном наборе компьютерного текста между двумя словами должен быть только один пробел.

Пример 9.9.

V. Программа:

```
var st: string;
   i: integer;
begin
 writeln('Введите строку');
 readln(st);
 i := 1;
 while i < length(st) do
 begin
  if (st[i] = ' ') and
     (st[i+1] = ' ') then
    delete(st, i, 1)
  else
   i := i + 1;
 end;
 writeln('Преобразованная
       строка: ');
 writeln(st);
end.
```

VI. Тестирование. Введите текст «При правильном наборе компьютерного текста между любыми двумя словами должен быть только один пробел».

Результат:

Окно вывода

```
Введите строку
При правильном  наборе компьютерного
текста между  любыми двумя словами
должен  быть только один  пробел.
Преобразованная строка:
При правильном наборе компьютерного
текста между любыми двумя словами
должен быть только один пробел.
```

словами должен быть только один пробел. Однако иногда случайно вставляют несколько пробелов. В этом случае Word подчеркивает их голубой волнистой линией (пример 9.8).

Пример 9.9. Написать программу, которая проверяет правильность расстановки пробелов в тексте и, если между словами более одного пробела, удаляет лишние.

Этапы выполнения задания

I. Исходные данные: переменная `st` (введенный текст).

II. Результат: преобразованный текст.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Поскольку длина строки будет изменяться при обработке строки, то количество повторений цикла заранее не известно. Будем использовать цикл **`while`**.

3. В цикле **`while`** проверяем соседние символы текста `st`.

3.1. Если оба соседних символа являются пробелами, то удалим один из них. Соседние символы имеют индексы, отличающиеся на один: `i` и `i + 1`. Поскольку в цикле есть обращение к элементу с номером `i + 1`, то условием выполнения цикла будет строгое неравенство `i < length(st)`.

3.2. Переходим к следующему символу только тогда, когда удаление не проводили.

4. Выводим результат.

IV. Описание переменных: `st` – `string`, `i` – `integer`.

Пример 9.10*. Написать программу, которая проверяет правильность расстановки пробелов вокруг тире. Если пробелы пропущены, то вставляет их. Предполагается, что в тексте нет слов, которые пишутся через дефис, двух знаков «–» подряд и лишних пробелов.

Этапы выполнения задания

I. Исходные данные: переменная `st` (введенный текст).

II. Результат: преобразованный текст.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Поскольку длина строки может изменяться при обработке строки, то количество повторений цикла заранее не известно. Будем использовать цикл **`while`**.

3. В цикле **`while`** проверяем каждый символ на совпадение с «–». При совпадении проверяем соседние символы.

3.1. Если соседний справа символ (`i + 1`) не пробел, то вставляем пробел.

3.2. Если символ слева (`i - 1`) не пробел, то вставляем пробел и увеличиваем `i` на 1.

3.3. Поскольку в цикле есть обращение к элементу с номером `i + 1`, то условием выполнения цикла будет строгое неравенство `i < length(st)`.

3.4. Поскольку в цикле есть обращение к элементу с номером `i – 1`, то начальное значение `i = 2`.

Пример 9.10*.

V. Программа:

```
var st: string;
    i: integer;
begin
  writeln('Введите строку');
  readln(st);
  i := 2;
  while i < length(st) do
  begin
    if st[i] = '-' then
    begin
      if st[i+1] <> ' ' then
        insert(' ',st, i+1);
      if st[i-1] <> ' ' then
      begin
        insert(' ',st, i);
        i := i + 1;
      end
    end;
    i := i + 1;
  end;
  writeln('Преобразованная строка: ');
  writeln(st);
end.
```

VI. Тестирование. Введите текст «Жизнь прожить–не поле перейти. Родимая сторона– мать, чужая –мачеха. Окончил дело – гуляй смело».

Результат:

Окно вывода

```
Введите строку
Жизнь прожить-не поле перейти.
Родимая сторона- мать, чужая -мачеха.
Окончил дело - гуляй смело.
Преобразованная строка:
Жизнь прожить - не поле перейти.
Родимая сторона - мать, чужая - мачеха.
Окончил дело - гуляй смело.
```

VII. Анализ результата. В примере встречаются все четыре возможные ситуации: пробелов нет ни слева, ни справа от тире; пробел только слева; пробел только справа; пробелы с двух сторон. В результате выполнения все пробелы расставлены правильно.

Пример 9.11*.
V. Программа:

```
var st, sl: string;
   p: boolean; n: integer;
procedure DelSpace(var s: string);
begin
 while (s<>'') and (s[1]= ' ') do
  delete(s, 1, 1);
end;
function FirstWord(s: string):
                 string;
var
 i: integer;
begin
 i := pos(' ', s);
 if i <> 0 then
  FirstWord := copy(s, 1, i - 1)
 else
  FirstWord := s;
end;
function CheckPalindrom
          (s: string): boolean;
var n: integer; f: boolean;
begin
 n := length(s); f := true;
 for var i := 1 to n div 2 do
  if s[i] <> s[n-i + 1] then
   f := false;
 CheckPalindrom := f;
end;
begin
 writeln('Введите st ');
 readln(st); p := false;
 while st <> '' do
 begin
  DelSpace(st);
  sl := FirstWord(st);
  if CheckPalindrom(sl) then
  begin
   writeln(sl); p := true;
  end;
  n := length(sl);
  delete(st, 1, n);
 end;
 if p = false then
  writeln('Нет палидромов');
end.
```

4. Переходим к следующему символу.

5. Вывод результата.

IV. Описание переменных: `st` – `string`, `i` – `integer`.

Пример 9.11*. Написать программу, которая выведет слова-палиндромы[1] (слова, которые одинаково читаются слева направо и справа налево), входящие в заданный текст. Слова в тексте могут быть разделены одним или несколькими пробелами. Пробелы могут быть в начале и в конце текста.

Этапы выполнения задания

I. Исходные данные: переменная `st` (введенный текст).

II. Результат: слова-палиндромы.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Будем выделять первое слово из исходного текста, проверять его и затем удалять.

3. Логическая переменная `p` изменит значение с `false` на `true`, если будет выведен палиндром.

4. Поскольку длина строки может изменяться при обработке строки, то количество повторений цикла заранее не известно. Будем использовать цикл **`while`**.

5. В цикле **`while`**, до тех пор, пока строка не станет пустой, выполняем следующее.

5.1. Удалим пробелы в начале строки.

5.2. Выделим первое слово из строки.

[1] В мире интересных слов. Палиндромы. http://www.tramvision.ru/words/pal.htm (дата доступа: 08.01.2019).

5.3. Проверим выделенное слово и, если оно является палиндромом, то выведем его.

5.4. Удалим слово из строки.

6. Опишем три вспомогательных алгоритма:

6.1. Процедуру `DelSpace` для удаления пробелов в начале строки. Пробелы удаляются из строки только тогда, когда она не пуста. Параметр данной функции будет изменяться внутри функции и должен остаться измененным после ее завершения, поэтому при описании перед параметром стоит ключевое слово **var**. Если пробелов в начале строки нет, то процедура не изменит исходную строку.

6.2. Функцию `FirstWord`, которая скопирует из строки первое слово. Если в строке только одно слово (нет пробелов), то оно же и является первым.

6.3. Функцию `CheckPalindrom` для проверки, является ли слово палиндромом. Будем сравнивать первый символ с последним, второй с предпоследним и т. д. Символ с номером `i` будет сравниваться с символом с номером (`n – i + 1`), где `n` — длина слова.

7. Вывод сообщения «Нет палиндромов» в случае, если значение `p` осталось `false`.

IV. Описание переменных: `st, sl` – `string`, `i, n` – `integer`, `p` – `boolean`.

Пример 9.11*. *Продолжение.*

VI. Тестирование. Введите текст:

«На берегу стоит шалаш из камыша. Для трафаретной печати предназначен мимеограф, или ротатор. Пшеничная лепешка наан является блюдом индийской национальной кухни».

Результат:

Окно вывода

```
Введите st
На берегу стоит шалаш из камыша.
Для трафаретной печати предназначен
мимеограф, или ротатор. Пшеничная
лепешка наан является блюдом индийской
национальной кухни.
шалаш
или
наан
```

VII. Анализ результатов. Из введенной строки не вывелось слово «ротатор», которое тоже является палиндромом. Но после этого слова стоит не пробел, а точка. Поэтому для проверки функция `CheckPalindrom` получает слово «ротатор.», которое не является палиндромом.

Чтобы в качестве палиндромов учитывались слова, после которых стоят знаки препинания, изменим функцию `CheckPalindrom`:

```
function
CheckPalindrom(s: string):
var n: integer;
   z: string;
   f: boolean;
begin
 n := length(s);
 z := ',.;:&!';
 if pos(s[n], z) <> 0 then
 begin
  delete (s, n, 1);
  n := n - 1;
 end;
 f := true;
 for var i := 1 to n div 2 do
  if s[i] <> s[n-i + 1] then
   f := false;
 CheckPalindrom := f;
end;
```

Пример 9.12. Примеры использования команд преобразования типов.

Преобразование числа к строковому представлению:

```
var a1: integer;
    a2: real;
    s1, s2: string;
begin
  a1 := 245; a2 := 3.7;
  s1 := IntToStr(a1);
  s2 := FloatToStr(a2);
  //выполним действия, чтобы
  //убедиться, что преобразование
  //типов произошло
  s1 := s1 + '1';
  writeln(s1);
  writeln(s2[2]);
end.
```

Результат:

Окно вывода

```
2451
.
```

Преобразование строкового представления числа к числовому значению:

```
var  a1:integer;
    a2: real;
    s1, s2: string;
begin
  s1 := '245'; s2 := '3.7';
  a1 := StrToInt(s1);
  a2 := StrToFloat(s2);
  //выполним действия, чтобы
  //убедиться, что преобразование
  //типов произошло
  a1 := a1 + 1;
  a2 := a2 + 0.1;
  writeln(a1, ' ', a2);
end.
```

Результат:

Окно вывода

```
246 3.8
```

При использовании процедур преобразования `Str(v,s)` и `Val(s,v,er)` тип числа определяется его записью.

Преобразование `Str(v,s)` возможно для любых доступных числовых типов.

9.3. Преобразование строк в числа и чисел в строки

Числовые данные используются для выполнения арифметических операций. Если символы цифр записаны в строковую переменную, то выполнять вычислительные действия с ними нельзя. Но можно преобразовывать строки, содержащие символы цифр, в числа и числа в строки, используя нижеперечисленные команды[1].

Команда	Описание
Функции преобразования числа a к строковому представлению	
`FloatToStr(a)`	Число a — вещественное
`IntToStr(a)`	Число a — целое
Функции преобразования строкового представления числа к числовому значению	
`StrToFloat(s)`	Строка s — запись вещественного числа
`StrToInt(s)`	Строка s — запись целого числа
Процедуры преобразования типов	
`Str(v,s);`	Преобразование числа в строку
`Val(s,v,er);`	Преобразование строки в число

Использование этих команд показано в примере 9.12.

В строковых переменных легко производить такие операции, как удаление, вставка или замена символа. Вставка, замена или удаление цифры

[1] Функции преобразования не работают с типом `BigInteger`.

из числа производятся сложнее. При необходимости число можно преобразовать в строку, выполнить необходимые действия и преобразовать строку обратно в число.

Пример 9.13. Написать программу, которая проверяет, является ли данный текст записью числа. В непустой текст могут входить только цифры или буквы. Если да, то найти сумму цифр данного числа, иначе вывести соответствующее сообщение.

Этапы выполнения задания

I. Исходные данные: переменная `st` (введенный текст).

II. Результат: сумма цифр или сообщение, что это не число.

III. Алгоритм решения задачи.

1. Вводим исходные данные.
2. Вычисляем длину строки.
3. К введенному тексту нельзя в явном виде применить функции преобразования типа, поскольку длина текста может быть больше 20, а простые числовые типы, содержащие такое количество цифр, Pascal не поддерживает. Поэтому будем пытаться преобразовывать в число каждый введенный символ, считать сумму и количество тех символов, которые удалось преобразовать.
4. Выполним инициализацию переменных: `s := 0` (сумма цифр числа) и `k := 0` (количество цифр).
5. В цикле **for** проверяем каждый символ строки `st`. Если текущий символ текста цифра, то преобразуем его в число, добавляем

Пример 9.12. *Продолжение.*

При использовании процедуры `Val(s,v,er)` сначала проверяется, возможно ли преобразование строковой записи в число в соответствии с типом. Если «да», то выполняется преобразование и переменная `er` получает значение 0 — код успешного преобразования, в противном случае значение `er` — это номер символа, который невозможно преобразовать. Вызов `Val('22.3',v,er)` присвоит переменной `v` значение 22.3, если она описана как `real` (`er = 0`); если она описана как `integer`, то `v` получит значение 22, `er = 3` (символ '.' не может быть преобразован).

Пример 9.13.

V. Программа:

```
var st: string; n, k, s: integer;
begin
  writeln('Введите текст');
  readln(st);
  n := length(st); k := 0; s := 0;
  for var i := 1 to n do
  begin
    if (st[i] >= '0') and
    (st[i] <= '9') then
    begin
      k := k + 1;
      s := s + StrToInt(st[i]);
    end;
  end;
  if k = n then
    writeln('Сумма цифр =', s)
  else
    writeln('Текст не число');
end.
```

VI. Тестирование. Введите текст 12345. Результат:

Окно вывода

```
Введите текст
12345
Сумма цифр = 15
```

Введите текст 123BC. Результат:

Окно вывода

```
Введите текст
123BC
Текст не число
```

Проверку того, что символ является цифрой, можно выполнить несколькими способами.

Если воспользоваться процедурой val, то фрагмент программы для проверки цифры будет таким:

```
for  var i := 1 to n do
 begin
  val(st[i], x, c);
  if c = 0 then
  begin
   k := k + 1;
   s := s + x;
  end;
 end;
```

Проверить, является ли символ цифрой, можно аналогично тому, как в примере 9.5 выполнялась проверка гласных букв. Для этого нужно создать строку, в которую перечислить все цифры — g:='0123456789';

Пример 9.14*.

V. Программа:

```
var st,sr,s1,s2,s3: string;
  a,b,c,n,r,r1,r2,p: integer;
begin
 writeln('Введите выражение');
 readln(st);
 sr := st + '=';
 //первое число a
 p := pos('(', st);
 s1 := copy (st, 1, p-1);
 a := StrToInt(s1);
 delete(st, 1, p);
 sr := sr + s1 + '*';
 //второе число b
 p := pos('+', st);
 s2 := copy (st, 1, p-1);
 b := StrToInt(s2);
 delete(st, 1, p);
 sr := sr + s2 + '+' + s1 +'*';
 //третье число c
 n := length (st);
```

число к сумме и увеличиваем счетчик количества преобразованных цифр. Если выполняется условие (st[i]>='0') **and** (st[i]<='9'), то символ строки является цифрой, поскольку символы цифр в таблице расположены последовательно.

6. Если количество символов, которые удалось преобразовать, равно длине строки, то выводим сумму цифр, иначе выводим соответствующее сообщение.

IV. Описание переменных: st – string, n, s, k – integer.

Пример 9.14*. Написать программу, которая раскрывает скобки в числовом выражении и вычисляет его значение. Выражение имеет вид $a(b + c)$ и вводится как строка. Вместо a, b и c — символы цифр, образующие целое число (количество цифр в каждом из них не более девяти). Вывести последовательность преобразований и результат. Например, для выражения 5(7 + 8) должны получить: 5(7 + 8) = = 5 * 7 + 5 * 8 = 35 + 40 = 75.

Этапы выполнения задания

I. Исходные данные: переменная st (текст).

II. Результат: числовое значение выражения.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Будем последовательно копировать из строки нужные символы и удалять те, которые уже обработали.

2.1. Находим символ «(», символы до него скопируем в переменную s1 и преобразуем ее в число a. Удалим эти символы из строки.

2.2. Находим символ «+», символы до него скопируем в переменную s2 и преобразуем ее в число b. Удалим эти символы из строки.

2.3. Из оставшейся строки скопируем в переменную s3 все символы, кроме последнего — «)», и преобразуем в число c.

3. Текущий результат будем добавлять к новой строке, которой вначале присваивается введенная строка и символ «=». Текущие результаты вычислений ($r1 := a*b$ и $r2 := a*c$) будем преобразовывать в строковый тип и добавлять к строке sr.

4. Вычисляем значение выражения.

5. Выводим результат.

IV. Описание переменных: st, sr, s1, s2, s3 — string, a, b, c, n, r, r1, r2, p — integer.

```
    s3 := copy (st, 1, n-1);
    c := StrToInt(s3);
    sr := sr + s3 + '=';
    //вычисление произведений
    r1 := a * b;
    sr := sr+IntToStr(r1)+'+';
    r2 := a * c;
    sr := sr+IntToStr(r2)+'=';
    //вычисление результата
    r := r1 + r2;
    sr := sr + IntToStr(r);
    writeln(sr);
end.
```

VI. Тестирование. Введите выражение 5(7 + 8).

Результат:

Окно вывода

```
Введите выражение
5(7+8)
5(7+8)=5*7+5*8=35+40=75
```

Введем выражение 12(234 + 802). Получим следующий результат:

Окно вывода

```
Введите выражение
12(234+802)
12(234+802)=12*234+12*802=2808+9624=12432
```

Упражнения

1 Напишите программу, которая определит количество предложений в тексте. Предложение заканчивается одним из трех символов: «.», «?», «!». Предполагается, что в тексте есть хотя бы одно предложение (см. пример 9.2).

2 Напишите программу, которая определит количество слов в тексте, если между любыми двумя словами может быть более одного пробела. Предполагается, что в тексте есть хотя бы одно слово (см. пример 9.2).

3 Напишите программу, которая определит, каких букв в строке с русским текстом больше: «о» или «О» (см. пример 9.3).

4 Напишите программу, которая определит, какой процент составляют буквы «а» во введенном тексте (см. пример 9.3).

5 Напишите программу, которая определит, сколько слов в тексте начинается на букву «а».

6* Напишите программу, которая определит, какой процент слов в тексте начинается на букву «к». (Слово может начинаться как с прописной, так и со строчной буквы.)

7 Дан текст. Напишите программу, которая проверит, правильно ли в нем расставлены круглые скобки. Если нет, то вывести соответствующее сообщение: «Открывающихся скобок больше (меньше), чем закрывающихся»; «Закрывающиеся скобки раньше открывающихся скобок».

8 В тексте могут встречаться гласные и согласные буквы, а также символы «ь» и «ъ». Измените программу из примера 9.5 так, чтобы символы «ь» и «ъ» выводились желтым цветом.

9 Дано арифметическое выражение, состоящее из цифр, скобок и знаков арифметических действий. Напишите программу, которая выведет цифры синим цветом, а остальные символы — голубым: например, в выражении 2 + (3 – 5) * 7 – 13 (см. пример 9.5).

10 Напишите программу для решения задачи. Задана строка цифр. Вывести четные цифры синим цветом, а нечетные — голубым (например, 128235). Сколько в строке нечетных цифр? (См. примеры 9.3 и 9.5.)

11* Вводится текст, слова в котором разделены пробелами, после слов могут стоять точки или запятые. Напишите программу, которая выведет синим цветом те буквы «а», которые являются последними буквами слова, остальные символы текста вывести голубым цветом (например, в скороговорке На дворе — трава, на траве — дрова). Какой процент от общего количества слов составляют слова, заканчивающиеся на букву «а»?

12 Напишите программу, которая заменит в заданном тексте каждую букву «а» символом «*» (см. пример 9.6).

13 Напишите программу, которая заменит в заданном тексте каждую цифру символом «?» (см. примеры 9.5 и 9.6).

14 Напишите программу, которая заменит в заданном тексте из латинских букв все вхождения «*x*» на «*ks*» (см. пример 9.7).

15 Напишите программу, которая заменит в заданном тексте из латинских букв все вхождения «*ing*» на «*ed*» (см. пример 9.7).

16* Напишите программу для решения задачи. В заданном тексте заменить все слова A1 на слова A2 (слова в тексте разделены пробелами, слова A1 и A2 вводятся).

17 Напишите программу, которая удалит из текста все гласные буквы (см. примеры 9.5 и 9.9).

18 Напишите программу, которая удалит из текста все знаки «+», непосредственно за которыми стоит не цифра.

19 Напишите программу, которая в заданном тексте после каждой латинской буквы «*q*» добавит букву «*u*» (см. пример 9.10).

20 Напишите программу, которая в заданном тексте после каждого знака препинания («.», «,», «:», «;») вставит пробел, если его там нет (см. пример 9.10).

21 Измените функцию `CheckPalindrom` из примера 9.11 так, чтобы слова, которые начинаются на заглавную букву, тоже считались палиндромами, например «Анна», «Алла».

22 Добавьте в программу из примера 9.11 подсчет количества выведенных палиндромов.

23* Фразы-палиндромы читаются одинаково слева направо и справа налево без учета пробелов и знаков препинания. Например: «Кулинар, храни лук» или «А роза упала на лапу Азора». Напишите программу, которая определит, является ли фраза палиндромом.

24 Напишите программу, которая проверяет, является ли данный текст записью числа. В непустой текст могут входить только цифры или буквы. Если да, то требуется проверить, делится ли данное число на 4, иначе вывести соответствующее сообщение. Для проверки делимости на 4 использовать признак делимости: число делится на 4, если двузначное число, состоящее из последних двух цифр исходного числа, делится на 4 (см. пример 9.13).

25 Измените программу из упражнения 20 так, чтобы проверялась делимость на 2, 3, 5, 6, 8, 12 (используйте соответствующие признаки делимости).

26 Дан текст. Напишите программу, которая проверит, может ли быть этот текст записью вещественного числа.

27 Напишите программу для решения задачи. Строка представляет собой запись следующего вида: «a ± *b*». Найти значение выражения. Вместо знака «±» может быть знак «+» или знак «–». Числа a и b являются целыми и состоят не более чем из девяти цифр (см. пример 9.14).

28 Напишите программу для решения задачи. Строка представляет собой запись следующего вида: «(*a* + *b*) / *c*». Выделить из записи числа и найти целочисленное значение выражения и остаток от деления. Числа, входящие в выражение, являются целыми и состоят не более чем из девяти цифр (см. пример 9.14).