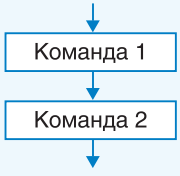
Глава 2
АЛГОРИТМЫ ОБРАБОТКИ СТРОКОВЫХ ВЕЛИЧИН

§ 6. Основные алгоритмические конструкции и типы данных

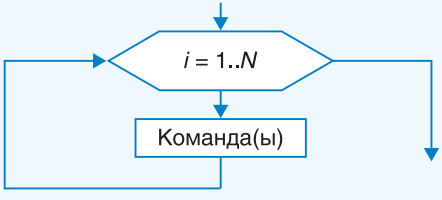
Пример 6.1. Блок-схемы алгоритмических конструкций.

1. Следование

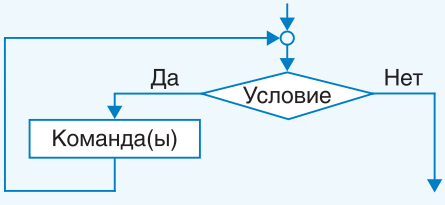


2. Цикл

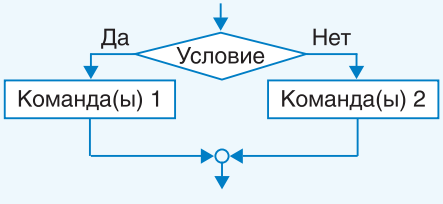
1) цикл с параметром



2) цикл с предусловием



3. Ветвление



6.1. Основные алгоритмические конструкции

Напомним некоторые определения, известные вам из курсов 7-го и 8-го классов.

Алгоритм — конечная последовательность команд, формальное выполнение которых позволяет получить решение задачи для любого допустимого набора исходных данных. Все команды делят на группы:

1. Команды, которые непосредственно выполняются в программе.
2. Команды, изменяющие порядок выполнения других команд.

Любой алгоритм может быть записан с использованием базовых алгоритмических конструкций, а именно: **следование, цикл** и **ветвление** (пример 6.1).

Программа представляет собой запись на некотором формальном языке — языке программирования. Командами в языке программирования считают:

- операторы (оператор присваивания, оператор ветвления, оператор цикла и др.);
- вызовы вспомогательных алгоритмов (встроенных в библиотеки и созданных пользователем).

Команды цикла и ветвления управляют порядком выполнения других команд в программе и относятся

к **командам управления (управляющим конструкциям)** (пример 6.2).

Оператор ветвления — команда, реализующая алгоритмическую конструкцию *ветвление* на языке программирования.

Оператор цикла — команда, реализующая алгоритмическую конструкцию *повторение* на языке программирования.

В Pascal существуют различные возможности управлять тем, сколько раз будет повторяться тело цикла. Может быть задано условие продолжения или окончания работы цикла, а также число повторений тела цикла.

Цикл с предусловием используется в том случае, когда известно условие продолжения работы.

Цикл с параметром используется тогда, когда известно количество повторений.

6.2. Вспомогательные алгоритмы

Вспомогательный алгоритм — алгоритм, который можно использовать в других алгоритмах, указав его имя и, если необходимо, значения параметров.

В языке Pascal используются вспомогательные алгоритмы двух видов: процедуры и функции. Они могут быть с параметрами или без параметров (пример 6.3).

Описание процедур и функций повторяет структуру программы на языке Pascal. Оно может содержать раздел **var** для описания переменных, которые используются только

Пример 6.2. Запись операторов ветвления и цикла на языке Pascal.

Ветвление в полной форме:

```
if <условие> then
begin
  команды 1;
end
else
begin
  команды 2;
end;
```

Ветвление в сокращенной форме

```
if <условие> then
begin
  команды;
end;
```

Цикл с предусловием

```
while <условие> do
begin
  тело цикла;
end;
```

Цикл с параметром

Параметр увеличивается:

```
for var i := N1 to N2 do
begin
  тело цикла;
end;
```

Параметр уменьшается:

```
for var i := N2 downto N1 do
begin
  тело цикла;
end;
```

Пример 6.3. Общий вид процедуры:

```
procedure <имя>(<список параметров>:тип);
var <описание переменных>
begin
  <команды>
end;
```

Общий вид функции:

```
function <имя>(<список параметров>:тип): тип результата;
var <описание переменных>
begin
  <команды>
  <имя> := <значение>;
end;
```

Функция должна содержать команду вида <имя> := <значение>;. Эта команда определяет, что функция должна вернуть в качестве результата.

Пример 6.4. Вызов процедуры и функции.

Вызов процедуры рисования круга:

```
Circle(250, 125, 30);
```

вызов функций для вычисления квадратного корня и синуса:

```
d := sqrt(2) * sin(x);
```

Пример 6.5. Целочисленные типы данных в PascalABC.

Тип	Диапазон значений	Размер памяти, байт
shortint	−128..127	1
smallint	−32768..32767	2
integer, longint	−2147483648.. 2147483647	4
byte	0..255	1
word	0..65535	2
longword, cardinal	0..4294967295	4

Дополнительно в PascalABC определен тип BigInteger, который не ограничен диапазоном значений и размером памяти.

Пример 6.6. Вещественные типы данных в PascalABC.

Тип	Диапазон значений	Размер памяти, байт
real (double)	$-1.8\cdot10^{308}..1.8\cdot10^{308}$	8
single	$-3.4\cdot10^{38}..3.4\cdot10^{38}$	4
decimal	$-(2^{96}-1)..2^{96}-1$	16

Количество значащих цифр в типе real составляет 15—16, в типе single — 7—8, в типе decimal — 28—29.

Тип real имеет другое название — double. Самое маленькое положительное число типа real приблизительно равно $5.0\cdot10^{-324}$, для типа single оно составляет приблизительно $1.4\cdot10^{-45}$.

внутри данной процедуры или данной функции.

Функции, в отличие от процедур, в результате своего выполнения возвращают значение, которое может быть использовано в выражении. Вызов процедуры является отдельной командой (пример 6.4).

6.3. Типы данных

В языке Pascal используются разные типы данных. Они нужны для выполнения различных операций — с каждым типом данных связан свой набор операций.

Для хранения различных типов данных в памяти компьютера отводится разное количество памяти. Вы уже использовали типы данных, которые называют простыми. В таблице примера 6.5 приведены целочисленные типы данных, используемые в PascalABC.

Все целочисленные типы данных, представленные в таблице, можно разделить на две группы:

- **знаковые** (диапазон значений которых содержит как положительные, так и отрицательные числа);
- **беззнаковые** (диапазон значений содержит только неотрицательные числа).

Для каждого знакового типа есть беззнаковый, занимающий столько же памяти.

Вещественные типы данных позволяют хранить число, представленное в стандартном виде: $a \cdot 10^n$, где $1 \leqslant a < 10$

и n (целое) есть порядок числа, записанного в стандартном виде (пример 6.6).

Значения типа boolean, который называют логическим, занимают 1 байт и принимают одно из двух значений, задаваемых константами true (истина) и false (ложь).

Данные в программу пользователь может вводить с помощью команды read (readln). Для вывода данных используется команда write (writeln).

Оператор присваивания используется для того, чтобы задавать значения переменным и вычислять значение выражения.

При использовании в одном операторе присваивания данных разных типов нужно помнить об их совместимости:

- переменной целочисленного типа нельзя присвоить вещественное значение;
- для данных вещественных типов определены операции «+», «−», «*», «/»;
- для данных целочисленных типов определены операции «+», «−», «*», «**div**», «**mod**».

Целочисленные типы могут быть преобразованы к вещественным, но не наоборот (пример 6.7).

Все простые типы, кроме вещественного, называются **порядковыми**. Значения только этих типов могут быть параметрами цикла **for**. Для порядковых типов используются функции ord, pred и succ, а также процедуры inc и dec (пример 6.8

Пример 6.7. Приведение типов.

```
var a, b, c: real;
    x, y, z: integer;
begin
  a := 1; b := 3; x := 6; y := 4;
  //вычисления с вещественным типом
  c := a / b; writeln(c);
  //преобразование к вещественному
  c := x / y; writeln(c);
  //вычисления с целым типом
  z := x div y; writeln(z);
  //преобразование к вещественному
  c := x div y; writeln(c);
end.
```

Результат:

Окно вывода

```
0.333333333333333
1.5
1
1
```

Пример 6.8. Процедуры и функции для работы с порядковыми типами.

Функция	Описание
Ord(a)	Порядковый номер значения a
Pred(x)	Значение, предшествующее x
Succ(x)	Значение, следующее за x

Процедура	Описание
Inc(i);	Увеличивает значение переменной i на 1
Inc(i, n)	Увеличивает значение переменной i на n
Dec(i);	Уменьшает значение переменной i на 1
Dec(i, n)	Уменьшает значение переменной i на n

Пример 6.9. Фрагмент программы для работы с порядковыми типами.

```
var a, c: integer;
    b: boolean;
begin
  a := 10; c := 3; b := true;
  writeln(ord(a));
  writeln(pred(b));
  writeln(succ(c));
  inc(b); writeln(b); dec(c, a);
  writeln(c);
end.
```

Окно вывода

```
10
False
4
False
-7
```

Пример 6.10.

V. Программа:

```
uses GraphABC;
const slovo = 'Pascal';
begin
  SetFontColor(clRed);
  writeln(slovo);
end.
```

VI. Тестирование.

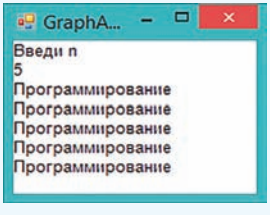
Запустить программу. Результат:

Пример 6.11.

V. Программа:

```
uses GraphABC;
const sl = 'Программирование';
var n: integer;
begin
  writeln ('Введи n ');
  read(n); writeln (n);
  for var i := 1 to n do
    writeln(sl);
end.
```

VI. Тестирование. Результат при $n = 5$:



и пример 6.9). Все переменные, которые используются в программе, должны быть описаны в разделе **var**. Если данные не изменяются в процессе работы программы, то они могут быть описаны как константы в разделе **const**. Например:

```
const slovo = 'Привет';
      Pi = 3.1416;
```

Для работы с графическими данными используются команды библиотеки GraphABC (см. *Приложение 2*, с. 159—160).

6.4. Примеры задач

Пример 6.10. Описать слово «Pascal» как константу. Вывести слово на экран красным цветом.

Этапы выполнения задания

I—II. Результат работы не зависит от исходных данных.

III. Алгоритм решения задачи.

1. Установить красный цвет (команда находится в библиотеке GraphABC).
2. Вывести константу.

IV. В программе нет переменных.

Пример 6.11. Написать программу, которая выведет заданное слово на экран n раз. Значение n вводится.

Этапы выполнения задания

I. Исходные данные: переменная n.

II. Результат: n слов.

III. Алгоритм решения задачи.

1. Описываем слово как константу.
2. Вводим значения n.
3. Воспользуемся циклом **for** для вывода слова n раз.
4. Выводим слова в цикле.

IV. Описание переменных: n — integer.

Пример 6.12. Написать программу, которая выведет на экран n раз одно из двух слов. Выбор слова осуществляется случайным образом. Значение n вводится. Посчитайте, сколько раз было выведено каждое слово.

Этапы выполнения задания

I. Исходные данные: переменная n.

II. Результат: n раз выведено одно из двух слов и сообщение о том, сколько раз выведено каждое из слов.

III. Алгоритм решения задачи.

1. Слова описываем как константы.

2. Вводим значения n.

3. Инициализируем нулем переменные k1 и k2, которые будут подсчитывать, сколько раз выведено каждое слово.

4. Для вывода слов используем цикл **for**.

4.1. Сгенерируем случайное число x на промежутке $[0; 2)$.

4.2. Если x = 0, то выведем первое слово и увеличим значение переменной k1 на 1.

4.3. Иначе выведем второе слово и увеличим значение переменной k2 на 1.

5. Выводим сообщения о количестве слов.

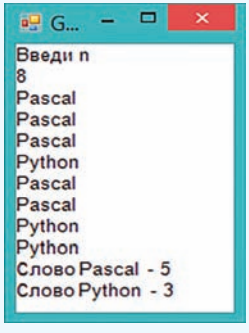
IV. Описание переменных: n, k1, k2, x — integer.

Пример 6.12.

V. Программа:

```
uses GraphABC;
const sl1 = 'Pascal';
      sl2 = 'Python';
var n, k1, k2, x: integer;
begin
  writeln ('Введи n ');
  read(n);
  writeln (n);
  k1 := 0; k2 := 0;
  for var i := 1 to n do
  begin
    x := random(2);
    if x = 0 then
    begin
      writeln(sl1);
      k1 := k1 + 1;
    end
    else
    begin
      writeln(sl2);
      k2 := k2 + 1;
    end
  end;
  writeln('Слово ',sl1,' - ',k1);
  writeln('Слово ',sl2,' - ',k2);
end.
```

VI. Тестирование. Запустить программу. Результат при $n = 8$:



1. Что такое алгоритм?
2. Назовите основные алгоритмические конструкции.
3. Что понимают под вспомогательным алгоритмом?
4. Чем отличаются различные целочисленные типы данных друг от друга?
5. Что нужно помнить о совместимости типов данных?
6. Какие арифметические операции определены для целочисленных типов данных? Для вещественных типов данных?

Упражнения

1 Измените константу в программе примера 6.10 на свое имя. Используя команды графической библиотеки для работы с текстом, измените шрифт, размер символов, фон, начертание букв.

2 Измените программу примера 6.11 так, чтобы выполнялись указанные условия.

1. Каждое слово должно выводиться случайным цветом.
2*. Расстояние между словами должно быть 50 пикселей.
3*. Каждое новое слово должно выводиться шрифтом на 5 пунктов больше, чем предыдущее. Отрегулируйте расстояние между словами так, чтобы слова при выводе не перекрывали друг друга.

3 Запустите программу из примера 6.12 несколько раз. Какие результаты получили?

4 Измените программу из примера 6.12 так, чтобы случайным образом выбиралось одно из трех слов. Выводите каждое слово своим цветом (например, первое — красным, второе — синим, третье — зеленым).

§ 7. Строковые величины

В первых языках программирования строкового типа данных не было; программист должен был сам строить функции для работы со строками.

В 1962 г. был разработан язык SNOBOL (StriNg Oriented symBOlic Language), ориентированный на работу со строками. В конце 60-х гг. XX в. строковые типы данных появились в языках Algol и Fortran.

Две строки, в отличие от двух чисел, нельзя прочитать с помощью одной команды `read`, поскольку пробел для строк не разделитель, а такой же символ, как и все остальные. Необходимо использовать две команды `readln`.

Если использовать две команды `read`, то первая строка будет считана так, как нужно, а вторая строка будет пустой (она не будет вводиться). Это происходит потому, что первая команда `read` считывает данные до нажатия клавиши Enter. Вторая команда `read` прочитает один символ — символ нажатия клавиши Enter.

7.1. Ввод, вывод, присваивание строковых величин

Современные компьютеры способны обрабатывать данные, представленные различными способами: числа, тексты, графику, звуки. Вы уже знаете, как на языке программирования Pascal можно работать с целыми и вещественными числами, выполнять простейшие графические построения. Обработка текстовых данных является сегодня наиболее актуальной — это обработка различных поисковых запросов в Интернете, перевод текстов с одного языка на другой, озвучивание компьютером печатного текста и др.

В языке Pascal для работы с текстовыми данными используется тип **`string` (строка).** Строки состоят из набора последовательно расположенных символов и используются для хранения текста. Они могут иметь произ-

вольную длину. Строка, в которой нет ни одного символа, называется **пустой**.

Строка описывается следующим образом:

```
var <имя строки>: string;
```

Для ввода и вывода строки используются те же команды, что и для ввода и вывода чисел: `read` (`readln`) и `write` (`writeln`) (пример 7.1). Ввод данных всегда заканчивается нажатием клавиши Enter, которой соответствует специальный символ: ¶. Команда `read` считывает символы в строку до тех пор, пока не встретится этот специальный символ. Команда `readln` отличается тем, что считывает не только данные в строку, но и символ нажатия Enter. Сам символ ¶ к строке не приписывается.

Переменной строкового типа можно присвоить значение с помощью команды присваивания. Значение строковой величины записывается в апострофах. Пустая строка задается следующим образом: `s := '';`

Запись поясняющего текста при выводе в команде `write` является строковой константой. К символам в строке можно обращаться, используя индекс. Нумерация символов начинается с единицы, `s[i]` соответствует `i`-му символу в строке `s` (пример 7.2).

Пример 7.3. Написать программу, которая спросит имя пользователя и выведет приветствие, обращаясь по имени.

Этапы выполнения задания

I. Исходные данные: `imja` — строка, в которой будет храниться введенное имя.

Пример 7.1. Введем строку *s* и выведем ее значение.

```
var s: string;
begin
  writeln('Введи строку');
  readln(s);
  writeln('Ввели строку: ', s);
end.
```

Результат работы:

Окно вывода

```
Введи строку
Pascal
Ввели строку: Pascal
```

Пример 7.2. Вывести третий символ строки «Информатика».

```
var s: string;
begin
  s := 'Информатика';
  writeln('3-й символ - ', s[3]);
end.
```

Результат работы:

Окно вывода

```
3-й символ - ф
```

Пример 7.3.

V. Программа:

```
var imja: string;
begin
  writeln ('Как тебя зовут?');
  readln(imja);
  writeln ('Привет, ',imja);
end.
```

VI. Тестирование.

Запустить программу и ввести значение «Катя». Результат:

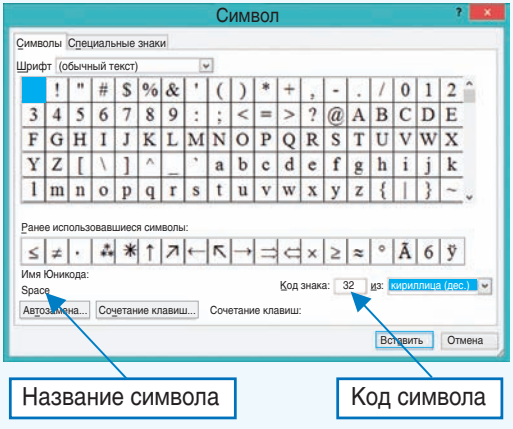
Окно вывода

```
Как тебя зовут?
Катя
Привет, Катя
```

Пример 7.4. Порядок расположения групп символов в таблице символов.

1. Пробел.
2. Символы цифр.
3. Заглавные латинские буквы.
4. Строчные латинские буквы.
5. Заглавные русские буквы.
6. Строчные русские буквы.

В нижеприведенной таблице символов выделен пробел, имеющий код 32. Далее видно расположение цифр и латинских букв.



Пример 7.5. Примеры сравнения строк.

S1	*S2*	Результат
'string'	'char'	*S1* > *S2*
'строка'	'Строка'	*S1* > *S2*
'книга'	'Журнал'	*S1* > *S2*
'мебель'	'стол'	*S1* < *S2*
'липа'	'лист'	*S1* < *S2*
'леска'	'лес'	*S1* > *S2*
'112'	'7'	*S1* < *S2*
'123'	'11111111'	*S1* > *S2*
'123'	'123'	*S1* = *S2*

II. Результат: строка с приветствием.
III. Алгоритм решения задачи.
1. Ввод имени.
2. Вывод строки с приветствием.
IV. Описание переменных: `imja` — `string`.

7.2. Сравнение и сложение строковых величин

Так же как и для других типов данных, для строк определены свои операции — действия, которые можно выполнять с данными типа `string`. Для строкового типа такими операциями являются сравнение и сложение.

Для сравнения строк нужно уметь сравнивать символы. Все символы записаны в таблице символов (в документе Word эту таблицу вы использовали для вставки символа, отсутствующего на клавиатуре). Каждый символ в таблице имеет свой код (номер), и при сравнении символов сравниваются их коды — номера в таблице символов.

В таблице символов, которую использует PascalABC, 65536 символов (стандарт Юникод — англ. *Unicod*[1]). Запомнить, в каком порядке записаны символы, невозможно. Достаточно знать, как расположены группы наиболее часто употребляемых символов. В примере 7.4 приведены некоторые группы символов в порядке их расположения в таблице.

Строки сравниваются посимвольно. Сначала сравниваются первые символы двух строк. Если символы различны, то больше та строка, символ которой имеет больший номер. Если

[1] https://ru.wikipedia.org/wiki/Юникод (дата доступа: 05.01.2019).

символы одинаковые, то переходят к сравнению следующих символов. Сравнение заканчивается, когда найдены различные символы или в одной из строк закончились символы — в этом случае больше та строка, в которой символы остались. Если при сравнении символов различия не найдены и строки закончились одновременно, то они равны (пример 7.5). Порядок, в котором меньшая строка предшествует большей, называют **лексикографическим**. Это название он получил по аналогии с размещением по алфавиту в словаре.

При сравнении строк следует помнить, что заглавные и строчные буквы — это разные буквы, поскольку имеют различные номера в таблице символов. Поэтому строки 'мама' и 'Мама' будут различными, а неравенство 'мама' > 'Мама' будет верным (у буквы 'м' код 236, а у 'М' — 204). Для проверки правильности сравнения строк можно воспользоваться программой из примера 7.6.

Пример 7.7. Написать программу, которая спросит имя пользователя, его пол, а затем поздоровается с ним, выводя красным цветом женские имена, а синим — мужские.

Этапы выполнения задания

I. Исходные данные: переменные `imja` (имя пользователя) и `pol` (пол пользователя).

II. Результат: строка с приветствием.

III. Алгоритм решения задачи.

Частично задача совпадает с примером 7.3, поэтому возьмем решение

Пример 7.6. Сравнение строк.

```
var S1,S2: string;
begin
 writeln('Введите первую строку');
  readln(S1);
 writeln('Введите вторую строку');
  readln(S2);
  if S1>S2 then
    writeln('S1 > S2')
  else
     if S1<S2 then
        writeln('S1 < S2')
     else
        writeln('S1 = S2')
end.
```

В PascalABC есть тип `char`, который позволяет хранить один символ. Над символами определены операции сравнения «<», «>», «<=», «>=», «=», «<>», которые сравнивают коды символов. Функция `ChrUnicode(n)` возвращает символ с кодом `n` (тип `char`), а функция `OrdUnicode(c)` — код символа (тип `byte`).

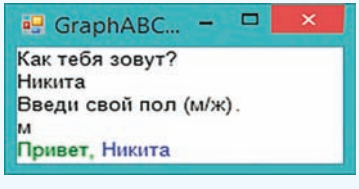
Пример 7.7.

V. Программа:

```
uses GraphABC;
var imja, pol: string;
begin
  writeln ('Как тебя зовут?');
  readln(imja); writeln(imja);
  writeln('Введи свой пол (м/ж).');
  readln(pol); writeln(pol);
  SetFontColor(clgreen);
  write('Привет, ');
  if pol = 'ж' then
    SetFontColor(clred)
  else
    SetFontColor(clblue);
  writeln (imja);
end.
```

VI. Тестирование.

Результат для имени Никита:



Пример 7.8. Примеры сложения строк.

```
var s1, s2, s3, s4: string;
begin
  s1 := 'Петя';
  s2 := 'Иванов';
  s3 := s1 + ' ' + s2;
  s4 := s2 + ' ' + s1;
  writeln(s3);
  writeln(s4);
end.
```

Результат:

```
Окно вывода
Петя Иванов
Иванов Петя
```

Пример 7.9.

V. Программа:

```
var str, gor, s: string;
begin
  writeln('Введите название государства');
  readln(str);
  writeln('Введите название столицы');
  readln(gor);
  s := 'Столица государства ' + str + ' – город ' + gor;
  writeln(s);
end.
```

VI. Тестирование.

Запустить программу и ввести значения Республика Беларусь и Минск. Результат:

```
Окно вывода
Введите название государства
Республика Беларусь
Введите название столицы
Минск
Столица государства Республика Беларусь – город Минск
```

из этого примера и изменим его. Цвет для текста можно задавать только в графическом окне, поэтому подключим графический режим.

1. Ввод исходных данных.

2. Сравнение переменной `pol` со значением `'ж'`. Если результат сравнения — истина, то устанавливаем красный цвет, иначе — синий.

3. Вывод строки.

IV. Описание переменных: `imja`, `pol` — `string`.

Для строк определена операция *сложение* (конкатенация). Обозначается операция знаком «+». В результате сложения двух строк получается новая строка, в которой после символов первой строки будут записаны символы второй строки. Результат данной операции зависит от порядка слагаемых (пример 7.8).

Пример 7.9. Написать программу, которая просит ввести название государства и его столицу, затем выводит сообщение:

Столица государства ... — город ... .

Вместо многоточия должны быть выведены соответствующие значения.

Этапы выполнения задания

I. Исходные данные: переменные `str` (название государства) и `gor` (название города).

II. Результат: переменная `s` (итоговая строка).

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Создание итоговой строки.

3. Вывод строки.

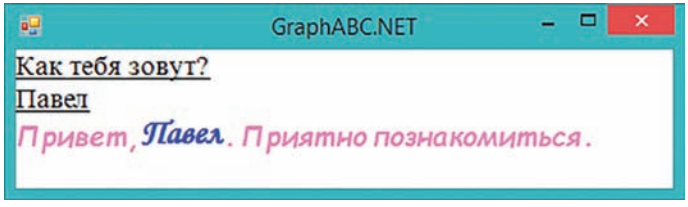
IV. Описание переменных: `str`, `gor`, `s` — `string`.

1. Как описываются строковые переменные?
2. Из чего состоят строки?
3. Какие операции возможны над строками?
4. Как упорядочены символы в таблице символов?
5. Как сравниваются строки?
6. Что является результатом сложения двух строк?

Упражнения

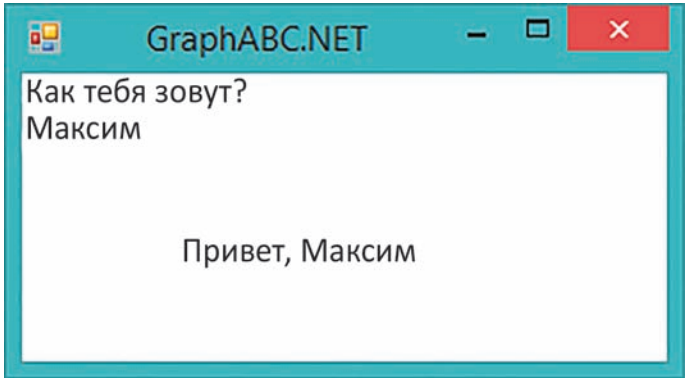
1 Внесите указанные изменения в программу из примера 7.3.

1. После имени пользователя выводить восклицательный знак.
2. Измените программу так, чтобы выводился текст:
Привет, Ольга. Приятно познакомиться.
3. Подключите графический режим. Задайте параметры шрифта для текста «Как тебя зовут?» и выводимого текста. Например, так:



2 Выполните следующие задания, изменив программу примера 7.3.

1. Выведите приветствие в графическом окне в точке с координатами (70; 70).



2. Задайте координаты вывода случайным образом.
3. Измените программу так, чтобы координаты месторасположения текста можно было вводить.
4. Задайте параметры шрифта для выводимых сообщений.

3 Используя программу из примера 7.6, проведите исследование по сравнению символов.

1. Совместно заполните таблицы.
2. Добавьте в каждую таблицу 2—3 строки с другими символами в соответствии с правилом сравнения.
3. Проверьте некоторые из результатов, используя таблицу символов (можно открыть в Word).

Сделайте выводы.

Сравнение символов русского алфавита, набранных в одном регистре		
S1	S2	Результат
ф	а	
С	Я	
я	у	

Сравнение символов латинского алфавита, набранных в одном регистре		
S1	S2	Результат
r	z	
W	J	
q	d	

Сравнение символов русского алфавита, набранных в разных регистрах		
S1	S2	Результат
ы	Ы	
а	Я	
я	А	

Сравнение символов латинского алфавита, набранных в разных регистрах		
S1	S2	Результат
Q	q	
Z	a	
A	z	

Сравнение символов цифр		
S1	S2	Результат
1	3	
7	9	
4	4	

Сравнение цифр и букв		
S1	S2	Результат
Q	1	
2	б	
9	ю	

Сравнение русских и латинских букв		
S1	S2	Результат
Q	л	
w	б	
Ч	k	

Сравнение знаков и цифр		
S1	S2	Результат
,	5	
7	!	
9	>	

Сравнение знаков и букв		
S1	S2	Результат
пробел	f	
W	{	
№	Я	

Сравнение знаков		
S1	S2	Результат
*	+	
.	,	
(	)	

4 Используя программу из примера 7.6, сравните строки. Откройте файл с таблицей и запишите результаты.

S1	S2	Результат сравнения
'string'	'String'	
'строка'	'символ'	
'alinéa'	'caractère'	
'Zeile'	'Zeichen'	
'cuerda'	'simbolo'	
'778'	'8'	
'876'	'55555555'	
'2 + 2'	'4'	

5 Какое приветствие выведет программа из примера 7.7, если, указывая пол, пользователь введет символ, отличный от «м» и «ж»? Внесите в программу изменения так, чтобы в этом случае вместо приветствия выводилось сообщение «Ошибка ввода».

6 Напишите программу, которая организует диалог с пользователем по следующему шаблону:

```
Как тебя зовут?
Ваня
Привет, Ваня. А ты любишь читать? (д/н)
д
Хорошо! Книга - источник знаний.
```

```
Как тебя зовут?
Петя
Привет, Петя. А ты любишь читать? (д/н)
н
Плохо, из книг можно узнать много интересного.
```

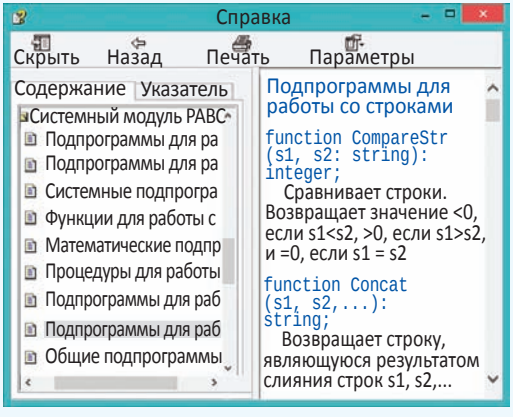
```
Как тебя зовут?
Леша
Привет, Леша. А ты любишь читать? (д/н)
v
Леша, ты ответил на вопрос некорректно.
```

Добавьте в программу свои вопросы и сообщения.

7 Напишите программу для решения следующей задачи: пользователь вводит свою фамилию, класс и учебное заведение, программа должна вывести сообщение «Учащийся ... учится в ... классе ГУО "..."». Вместо многоточия должны быть выведены соответствующие значения.

§ 8. Стандартные процедуры и функции для работы со строковыми величинами

Пример 8.1. В справке среды программирования PascalABC.NET в разделе **Системный модуль PascalABC-System → Подпрограммы для работы со строками** можно найти описание функций и процедур:

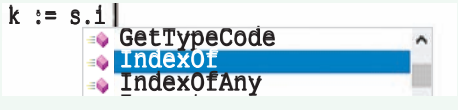


Пример 8.2. Примеры использования функций.

`d := Length ('Компьютер');`	d = 9
`s := 'Строка';` `d := Length(s);`	d = 6

`s := 'Не слово хозяин хозяину, а хозяин слову хозяин';`
`s1 := 'хозяин';`
`d := Length(s);`
`N1 := Pos(s1,s);`
`N2 := LastPos(s1,s);`
`N3 := PosEx(s1,s,15);`
	d = 46
	N1 = 10
	N2 = 41
	N3 = 17

PascalABC позволяет обращаться к функциям обработки строк и по-другому: если после имени строковой переменной поставить точку, то появится список функций.



8.1. Поиск в строке

Современные компьютерные устройства позволяют достаточно быстро осуществлять поиск в тексте, используя для этого различные алгоритмы. Языки программирования предоставляют широкий набор функций для работы с текстом. Некоторые функции языка программирования Pascal для поиска подстроки (части строки) в другой строке представлены в таблице.

Функция	Описание
`Length(s)`	Определяет длину строки `s` (количество символов в строке)
`Pos(s1, s)`	Определяет позицию подстроки `s1` в строке `s`. Если не найдена — возвращает 0
`LastPos (s1, s)`	Определяет позицию последнего вхождения подстроки `s1` в строке `s`. Если не найдена — возвращает 0
`PosEx (s1,s,from)`	Определяет позицию подстроки `s1` в строке `s`, начиная с позиции `from`. Если не найдена — возвращает 0

Подробное описание функций и процедур для работы со строками можно найти в справочной системе PascalABC.NET (пример 8.1), а также в *Приложении 2* (см. с. 161—162).

В примере 8.2. показано, как применять указанные функции.

Пример 8.3. Написать программу, которая вводит слово, а затем выводит его по одному символу в строке.

Этапы выполнения задания

I. Исходные данные: переменная s — исходное слово.

II. Результат: слово на экране, каждый символ в отдельной строке.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Определяем длину слова. Переменной n присваиваем значение функции length(s).

3. В цикле **for** выводим по одному символы введенного слова.

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

Пример 8.4. Написать программу, которая выводит на экран последний символ введенного слова и определяет, встречается ли этот символ в слове еще раз. Если встречается, то программа выводит индекс символа.

Этапы выполнения задания

I. Исходные данные: переменная s — введенное слово.

II. Результат: последний символ в слове и соответствующее сообщение — встречается или не встречается.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Определяем длину слова. Переменной n присваиваем значение функции length(s).

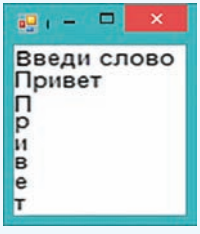
3. Определяем последний символ. Поскольку символы в строке нумеруются с 1, то номер последнего символа совпадает с длиной строки.

Пример 8.3.

V. Программа:

```
uses GraphABC;
var
  s: string; n: integer;
begin
  writeln('Введи слово');
  readln(s); writeln(s);
  n := length(s);
  for var i := 1 to n do
    writeln(s[i]);
end.
```

VI. Тестирование.



Пример 8.4.

V. Программа:

```
var s: string; n, k: integer;
begin
  writeln('Введи слово');
  readln(s); n := length(s);
  writeln('Последний символ - ',
s[n]);
  k := pos(s[n], s);
  if k = n then
    writeln('Символ один')
  else
   writeln('Символ с индексом ', k)
end.
```

VI. Тестирование. Результаты:

Окно вывода
Введи слово Строка Последний символ - а Символ один

Окно вывода
Введи слово Информатика Последний символ - а Символ на месте 7

Один из самых известных и эффективных алгоритмов поиска подстроки в строке — алгоритм Кнута — Морриса — Пратта (КМП-алгоритм). Алгоритм был разработан Д. Кнутом и В. Праттом и (независимо от них) Д. Моррисом. Результаты своей работы они опубликовали совместно в 1977 г. Время работы алгоритма линейно зависит от объема входных данных.

Дональд Кнут (род. в 1938)

Вон Пратт (род. в 1944)

Джеймс Моррис (род. в 1941)

Дональд Эрвин Кнут — американский ученый, автор известной серии книг об основных алгоритмах и методах вычислительной математики.

Вон Рональд Пратт — почетный профессор в Стэнфордском университете, известен своим вкладом в развитие таких областей информатики, как алгоритмы поиска и сортировок, а также тестирование простоты чисел.

Джеймс Хирэм Моррис — американский профессор. Занимался разработкой в области языков программирования и технологическим дизайном программных продуктов.

Пример 8.5. Примеры использования функций.

```
s := 'Информатика';
s := UpperCase(s);
```
После преобразования в строке `s` будет записано `ИНФОРМАТИКА`

```
s := 'Информатика';
s[1] := LowCase(s[1]);
```
После преобразования в строке `s` будет записано `информатика`

4. Определяем позицию последнего символа в слове. Переменной `k` присваиваем значение функции `pos` для последнего символа. Если оно равно длине строки, то символ в слове единственный, иначе в слове есть другой такой же символ.

5. Выводим результат.

IV. Описание переменных: `s` — `string`, `n`, `k` — `integer`.

Поиск текстовой информации во всемирной паутине заключается в том, чтобы по запросу пользователя найти документы, содержащие указанные ключевые слова. Для этого различные поисковые системы используют разные алгоритмы. Запрос, который вводит пользователь, может содержать строчные и заглавные буквы. Для осуществления поиска буквы в слове обычно приводят к одному регистру: либо все строчные, либо все заглавные. В Pascal также есть функции преобразования.

Функция	Описание
`LowCase(c)`	Преобразует один символ (букву) в строчную букву
`LowerCase(s)`	Преобразует все символы (буквы) строки в строчные буквы
`UpCase(c)`	Преобразует один символ (букву) в заглавную букву
`UpperCase(s)`	Преобразует все символы (буквы) строки в заглавные буквы

В примере 8.5 показано применение этих функций.

8.2. Копирование, вставка и удаление символов

При работе с текстом в текстовом редакторе часто приходится пользоваться буфером обмена. Часть текста (подстрока) копируется (вырезается) в буфер обмена, а затем вставляется в другое место в тексте. В языке Pascal реализованы команды для работы с фрагментом текста, которые представлены в таблице.

Команда	Описание
`Copy (s,index,count)`	Функция копирует часть строки `s` в другую строку
`Delete (s,index,count);`	Процедура удаляет символы строки `s`
`Insert (s1,s,index);`	Процедура вставляет подстроку `s1` в строку `s`

Во всех командах переменная `s` обозначает исходную строку, над которой производится операция. Переменная `index` обозначает позицию символа, начиная с которого выполняют операцию, а переменная `count` — количество символов. Разберем команды подробнее (пример 8.6).

Команда `copy` является функцией, и результат ее работы присваивается другой переменной. Строка `s` при этом не изменяется.

```
s1 := copy(s, index, count);
```

Команды `delete` и `insert` являются процедурами, они изменяют строку `s`. В примере 8.7 показано, как применяются указанные команды.

Пример 8.6. Команды для преобразования строк.

1. Запись `s1 := Copy(s, index, count);` означает, что в строке `s` выделяют `count` символов, выделение начинают с символа, индекс которого записан в переменной `index`. Эти символы копируются в переменную `s1` (действие команды сравнимо с копированием фрагмента текста в буфер обмена).

2. Запись `Delete(s, index, count);` означает, что в строке `s` выделяют `count` символов, выделение начинают с символа, индекс которого записан в переменной `index`. Выделенные символы удаляются из строки `s`. Остальные символы строки сдвигаются влево (действие команды сравнимо с удалением фрагмента текста).

3. Запись `Insert(s1, s, index);` означает, что в строку `s` вставляют символы строки `s1`, вставка происходит в позиции `index`. Остальные символы строки сдвигаются вправо (действие команды сравнимо со вставкой фрагмента текста из буфера обмена).

Пример 8.7. Примеры использования команд.

`s := 'Информатика';`	
`s1 := copy(s,3,5);`	`s1 = 'форма'`
`Delete(s,8,4);`	`s = 'Информа'`
`Insert ('ция',s,8);`	`s = 'Информация'`

Пример 8.8.

V. Программа:

```
var  s, p, t: string;
     n1, n2, k: integer;
begin
  writeln('Строка s');
  readln(s);
  writeln('Подстрока p');
  readln(p);
  n1 := length(s);
  n2 := length(p);
  k := 0;
  for var i := 1 to n1 - n2 + 1 do
  begin
    t := copy(s, i, n2);
    if t = p then
      k := k + 1;
  end;
  writeln('Встречается ', k,
' раз(-а)');
  end.
```

VI. Тестирование.

Запустить программу, ввести строку «Не слово хозяин хозяину, а хозяин слову хозяин» и подстроку «хозяин». Результат:

Окно вывода

```
Строка s
Не слово хозяин хозяину, а хозяин слову хозяин
Подстрока p
хозяин
Встречается 4 раз(-а)
```

Если для той же строки ввести подстроку «хозяйка», то результат будет таким:

Окно вывода

```
Строка s
Не слово хозяин хозяину, а хозяин слову хозяин
Подстрока p
хозяйка
Встречается 0 раз(-а)
```

Пример 8.8. Написать программу, которая определит, сколько раз заданная подстрока встречается в строке.

Этапы выполнения задания

I. Исходные данные: переменная s — исходная строка, p — исходная подстрока.

II. Результат: k — искомое количество.

III. Алгоритм решения задачи.

1. Вводим исходные данные.

2. Инициализируем значение счетчика k := 0;

3. Определяем длины n1 и n2 для строки s и подстроки p.

4. В цикле **for** от 1 до разницы в длинах строки s и подстроки p:

4.1. Выделяем из строки s подстроку t такой же длины, что и длина p, начиная с текущего символа.

4.2. Сравниваем подстроки. Если они равны, то увеличиваем значение счетчика на 1.

5. Выводим результат.

IV. Описание переменных: s, p, t – string, n1, n2, k – integer.

Пример 8.9. Написать программу, которая из слова ТЕСТИРОВАНИЕ получит слово РИСОВАНИЕ, используя процедуры и функции преобразования строк.

Этапы выполнения задания

I. Исходные данные: слово ТЕСТИРОВАНИЕ будем хранить как константу с именем s.

II. Результат: полученное слово.

III. Алгоритм решения задачи.

1. Результат не зависит от вводимых данных.

2. В строку S1 запишем шестой символ исходной строки.

3. Скопируем из строки S восемь символов, начиная с позиции 5. Добавим к строке S1. Получим 'РИРОВАНИЕ'.

4. В полученной строке удалим третий символ ('РИОВАНИЕ').

5. Вставим на третье место третий символ исходной строки ('РИСОВАНИЕ').

6. Выведем результат.

IV. Описание переменных: S1 — string.

Пример 8.9.

V. Программа:

```
const S = 'ТЕСТИРОВАНИЕ';
var S1: string;
begin
 S1 := S[6]; S1 := S1 + copy(S,5,8);
 //РИРОВАНИЕ
 delete(S1, 3, 1);
 //РИОВАНИЕ
 insert(S[3], S1, 3);
 //РИСОВАНИЕ
 writeln('Слово =', S1);
end.
```

VI. Тестирование.

Окно вывода

```
Слово = РИСОВАНИЕ
```

Для копирования последних 8 символов из строки S можно использовать функцию RightStr: S1 := S1 + RightStr(S,8).

Для копирования первых символов из строки можно использовать функцию LeftStr.

1. Что такое длина строки?
2. С помощью какой функции можно найти длину строки?
3. Какие функции используются для определения позиции подстроки в строке?
4. Как скопировать символы из одной строки в другую?
5. Какая процедура используется для удаления символов из строки?
6. Какая процедура предназначена для вставки символов в строку?
7*. Сопоставьте команды преобразования строк с операциями, выполняемыми с помощью буфера обмена.

Упражнения

1 В программу из примера 8.3 внесли следующие изменения:

```
for var i := 1 to n do
 begin
  write(s[i]);
  if i mod 2 = 0 then
   writeln;
 end;
```

Как теперь выводится слово? Объясните почему.

2 Измените программу из примера 8.3 так, как указано ниже.

1. Каждая буква должна выводиться своим цветом (можно использовать случайное задание цветов).

2. Буквы, стоящие на четных местах, должны выводиться одним цветом, а на нечетных — другим.

3 Измените программу из примера 8.4 так, чтобы на экран выводился символ введенного слова, стоящий посередине (для слов с четным количеством букв — символ справа от середины).

1. Проверьте правильность работы своей программы на предложенных примерах. Откройте файл с таблицей и запишите результаты.

Слово	Результат
Школа	о
гимназия	а
форма	
Интернет	

2. Допишите в таблицу два своих примера.
3. Что будет выведено, если ничего не вводить, просто нажать Enter?
4. Проверьте, встречается ли выведенный символ в слове еще раз.
5. Выведите позиции всех символов, совпадающих с символом слова, находящимся посередине.

4 Измените программу из примера 8.4 так, чтобы строчные и заглавные буквы анализировались программой одинаково (например, для слова «Анна» ответ должен быть следующим: «`Последний символ – а, символ встретился на месте 1`»).

5 Даны два слова. Верно ли, что одно из слов начинается на ту же букву, на которую заканчивается другое? (Первая буква одного из слов может быть заглавной.) Если да, то вывести букву, иначе — соответствующее сообщение.

1. Проверьте правильность работы своей программы на предложенных примерах. Откройте файл с таблицей и запишите результаты.

Слово 1	Слово 2	Результат
array	yellow	y
apple	auto	неверно
Рыба	Арбуз	

2. Допишите в таблицу два своих значения.
3*. Если ответ «верно», указать, принадлежат ли буквы одному регистру.

6 Измените программу из примера 8.8 так, чтобы при `k = 0` выводилось сообщение `'Подстрока в строке не встречается'`.

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.

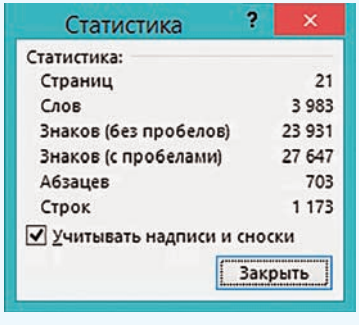
§ 9. Составление алгоритмов обработки строковых величин

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:

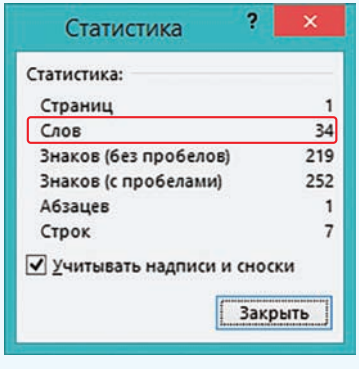
```
var st: string;
    n, k: integer;
begin
  writeln('Введите текст');
  readln(st);
  n := length(st);
  k := 1;
  for var i := 1 to n do
  begin
    if st[i] = ' ' then
      k := k + 1;
  end;
  writeln('В тексте ', k, '
    слов(-о/-а)');
end.
```

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
```

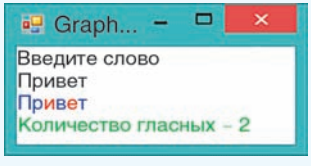
Пример 9.4. Русский алфавит.

Пример 9.5.

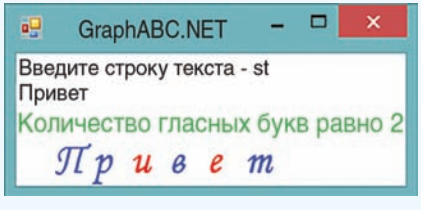
V. Программа:

```
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).