Кафедра конструирования и производства радиоаппаратуры

УТВЕРЖДАЮ
Заведующий кафедрой КИПР
_____В.Н. ТАТАРИНОВ
“___” _____2012 г.

Формирование текстов

Лабораторная работа по дисциплинам ***«***Информатика***»*** для студентов специальностей 211000.62 «Конструирование и технология электронных средств» (бакалавриат) и 162107.65 «Информатика и информационные технологии» (специалитет)

Разработчик:
Доцент кафедры КИПР
_____Ю.П. Кобрин

Томск 2012

Оглавление

1 Цели работы

- Изучение правил объявления и использования строковых и символьных данных в ***Borland Pascal***.
- Освоение алгоритмов и программ обработки символьных и строковых данных в ***Borland Pascal***.
- Знакомство с возможностями применения множеств в алгоритмах и программах на ***Borland Pascal***.
- Получение практических навыков в использовании встроенных подпрограмм преобразования символьной информации в ***Borland Pascal***.

2 Порядок выполнения работы

В ходе выполнения этой работы следует:

1) Изучить описание лабораторной работы, обратив особое внимание на возможности встроенных подпрограмм по преобразованию строковых и символьных данных в ***Borland Pascal***. При необходимости использовать дополнительную литературу [1 - 8].
2) Ответить письменно на контрольные вопросы.
3) Войти в свой личный каталог, и настроить интегрированную среду ***Borland Pascal*** для последующей работы. Записать файл конфигурации в личный каталог.
4) Получить вариант задания у преподавателя.
5) Внимательно проштудировать учебный пример и по возможности максимально использовать его при составлении своей задачи.
6) Разработать, ввести и отладить программу в соответствии со своим вариантом задания.
7) Продемонстрировать работоспособность программы для различных вариантов исходных данных.
8) Оформить отчет по лабораторной работе и защитить его у преподавателя.

3 Контрольные вопросы

Ответьте письменно на следующие контрольные вопросы:

1) Что такое множество, как оно описывается на языке ***Borland Pascal***?
2) Какие типы данных могут использоваться в качестве базового типа при объявлении типа множества в ***Borland Pascal***?
3) Какие операции возможны над множествами и каков их приоритет?
4) Как кодируют символы в ***Borland Pascal***?
5) Как получить значение кода требуемого символа?
6) Что представляет собой строка символов?
7) Как описываются строковые переменные?
8) Какая максимальная длина строки символов допустима в ***Borland Pascal***?
9) Какие операции допустимы над строковыми данными?
10) В чем отличие строковой переменной от массива символов?
11) Перечислите встроенные подпрограммы для обработки строк символов и охарактеризуйте их назначение.
12) Как можно выделить слово из строки символов?

4 Отчетность

Отчет должен быть выполнен в соответствии с [9] и состоять из следующих разделов:

1) Тема и цель работы.
2) Индивидуальное задание.
3) Схема алгоритма решения задачи.
4) Текст программы и вводимые тестовые исходные данные.
5) Откомпилированный текст программы-заготовки (в электронном виде).
6) Результаты выполнения программы.
7) Ответы на контрольные вопросы.
8) Выводы.

При защите отчета по работе для получения зачета студент должен:

- *уметь* отвечать *на контрольные вопросы*;
- обосновать структуру выбранного алгоритма и показать его работоспособность;
- уметь пояснять работу программы;
- продемонстрировать *навыки работы в среде* ***Borland Pascal.***

5 Работа с символьными и строковыми типами данных и множествами

5.1 Символьная информация

Во многих программах компьютеры помимо числовых данных обрабатывают также и символьные данные, которые могут быть представлены отдельными символами или строками символов.

Символьная информация – это информация, отображаемая с помощью символов (букв, цифр, знаков операций и др.). Каждый символ в памяти занимает один байт, и ему присваивается порядковый номер в диапазоне 0 .. 255. Соответствие символов и байтов задается *таблицей кодировки*, в которой для каждого символа указывается соответствующий код (порядковый номер).

Символы с кодами от 0 до 127 построены по стандарту ***ASCII*** (*Табл. 5.1 - American Standard Code for Information Interchange* — Американский стандартный код обмена информацией, читается "аски"). Вторая половина таблицы (коды 128 ... 255 - *Табл. 5.2*) в нашей стране содержит русские буквы (кириллицу) и символы псевдографики.

Таблица 5.1 – Кодировка символов в соответствии со стандартом ASCII

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0	NUL	32	BL	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j

Код	Символ	Код	Символ	Код	Символ	Код	Символ
11	VT	43	+	75	K	107	k
12	FF	44	/	76	L	108	1
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	0	111	o
16	DEL	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	\
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	⌂

Таблица 5.2 – Кодировка символов в соответствии со стандартами России (модифицированный альтернативный вариант)

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	А	160	а	192	└	224	р
129	Б	161	б	193	┴	225	с
130	В	162	в	194	┬	226	т
131	Г	163	г	195	├	227	у
132	Д	164	д	196	─	228	ф
133	Е	165	е	197	┼	229	х
134	Ж	166	ж	198	╞	230	ц
135	З	167	з	199	╟	231	ч
136	И	168	и	200	╚	232	ш
137	Й	169	й	201	╔	233	щ
138	К	170	к	202	╩	234	ъ
139	Л	171	л	203	╦	235	ы
140	М	172	м	204	╠	236	ь
141	Н	173	н	205	═	237	э
142	О	174	о	206	╬	238	ю
143	П	175	п	207	╧	239	я
144	Р	176	▒	208	╨	240	Ё
145	С	177	▒	209	╤	241	ё
146	Т	178	▓	210	╥	242	Є
147	У	179	│	211	╙	243	є
148	Ф	180	┤	212	╘	244	Ї
149	Х	181	╡	213	╒	245	ї

Код	Символ	Код	Символ	Код	Символ	Код	Символ
150	Ц	182	╢	214	╓	246	Ў
151	Ч	183	╖	215	╫	247	ў
152	Ш	184	╕	216	╪	248	°
153	Щ	185	╣	217	┘	249	∙
154	Ъ	186	║	218	┌	250	·
155	Ы	187	╗	219	█	251	√
156	Ь	188	╝	220	▄	252	№
157	Э	189	╜	221	▌	253	¤
158	Ю	190	╛	222	▐	254	■
159	Я	191	┐	223	▀	255	

Символы с кодами 0..31 служебные (*Табл. 3.3*) и участвуют в операциях ввода-вывода. На экране они обычно отображаются пробелами[1].

Таблица 5.3 – Назначение некоторых служебных символов

Символ	Код	Значение
BEL	**7**	*Звонок*; вывод на экран этого символа сопровождается звуковым сигналом
HT	**9**	Горизонтальная табуляция; при выводе на экран смещает курсор в позицию, кратную 8, плюс 1 (9, 17, 25 и т.д.)
LF	**10**	Перевод строки; при выводе его на экран все последующие символы будут выводиться, начиная с той же позиции, но на следующей строке
VT	**11**	Вертикальная табуляция; при выводе на экран заменяется специальным знаком
FF	**12**	Прогон страницы; при выводе на принтер формирует страницу, при выводе на экран заменяется специальным знаком
CR	**13**	Возврат каретки; вводится нажатием на клавишу Enter (при вводе с помощью READ или READLN означает команду «Ввод» и в буфер ввода не помещается; при выводе означает команду «Продолжить вывод с начала текущей строки»)
SUB	**26**	Конец файла; вводится с клавиатуры нажатием Ctrl-2; при выводе заменяется специальным знаком
ESC	**27**	Конец работы; вводится с клавиатуры нажатием на клавишу ESC; при выводе заменяется специальным знаком

Внимание! Для хранения и обработки отдельных символов используются константы и переменные типа ***Char***.

Значение символьной переменной или константы - это один символ из допустимого набора (см. *табл. 5.1 – 5.3*). Изменить значение переменной типа ***Char*** можно выполнением *операторов присваивания* или *ввода*.

Символьные константы могут записываться в тексте программы двумя способами:

1) как один символ, заключенный в апострофы, например:

[1] Условимся далее обозначать пробел (если на это нужно обратить внимание специально) скобочкой ∪.

'A' 'a' 'Ю' 'ю';

2) с помощью конструкции вида #*K*, где *K* - порядкового номер (код) соответствующего символа, при этом значение *K* должно находиться в пределах 0..255. Очень удобно представлять так служебные символы (см. *табл. 5.3*), для которых на клавиатуре клавиши не предусмотрены, например:

#10#13#7

К величинам символьного типа применимы *все операции отношения*.

С помощью функции ***Ord(Ch)*** можно узнать порядковый номер (код внутреннего представления) у любого символа ***Ch***. Функция ***Chr(K)*** определяет по порядковому номеру ***K*** символ, стоящий на *K*-ом месте в наборе символов. Порядковый номер имеет целый тип.

К аргументам символьного типа применяются функции ***Pred(Ch)*** и ***Succ(Ch),*** которые определяют предыдущий и последующий символы, например:

Pred('F') = 'E' ; Succ('Y') = 'Z' .

При отсутствии предыдущего или последующего символов значение соответствующих функций не определено.

Для литер из интервала ***'a'..'z'*** применима функция ***UpCase(Ch),*** которая переводит эти литеры в верхний регистр ***'A'..'Z'.***

Перед использованием символьные переменные следует описать:

```
Var
   <Идентификатор переменной, …>: char;
```

Например,

```
Var
  Otvet, Ch1, Ch2: char; {объявляются три символьные переменные}
Begin
  Ch1 := '+';          {присваивается символ плюс}
  Ch2 := #7;           {присваивается служебный символ короткого звукового сигнала}
  Readln(Otvet)        {вводится символ с клавиатуры}
end.
```

5.2 Строки символов

Внимание! Для хранения и обработки последовательностей (строк) символов в ***Pascal*** используются константы и переменные типа ***String***.

Строка символов *(строковая константа)* это последовательность символов (см. *табл. 5.1 – 5.3*), записанная в одной строке программы и заключенная в одиночные кавычки (апострофы).

Учтите! Вся последовательность символов должна располагаться в одной строке программы.

Чтобы включить в состав строки символ апострофа, следует этот символ отпечатать два раза. Строка символов, ничего не содержащая между апострофами **''**, называется *нулевой (пустой) строкой*.

Строковую переменную можно задать следующим образом:

Type
<Идентификатор типа строки>: *string*[фиксированная длина строки];
Var
<Идентификатор переменной, …>: <Идентификатор типа строки>;

или

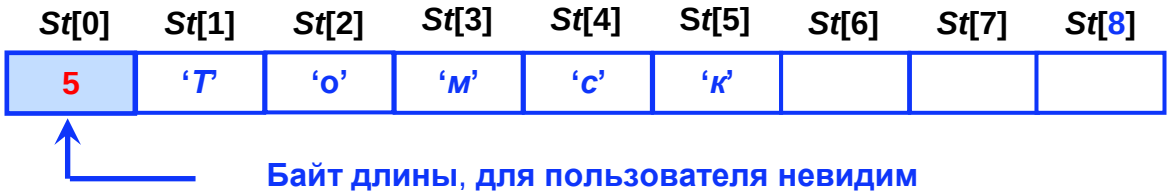
Var
<Идентификатор переменной, …>: *string*[фиксированная длина строки];

Например,

```
Var
  St1 : string[80];  {длина строки - 80 символов}
  St2: string[255]; {максимально-допустимая длина строки}
  St3 : string;      {по умолчанию длина строки максимальная - 255 символов}
```

В памяти компьютера строка представляет особую форму одномерного массива и занимает количество байтов, на единицу большее ее фиксированной длины. Нулевой байт строки содержит ее длину (*рис. 5.1*). Первый байт содержит текущую динамическую длину строки, а последующие байты содержат символы строки. Динамическая длина символьной строки (ее можно узнать с помощью стандартной функции ***Length***) - это фактическое число символов в строке.

```
St := 'Томск';
```



8 – фиксированная общая длина строки (переменная типа *String[8]*)
5 – динамическая (текущая) длина строки

Рисунок 5.1- Представление строки символов

Особенностью строковых переменных является то, что к ним можно обращаться как к скалярным переменным, так и к массивам. Изменить значение строчной переменной можно либо путем присвоения нового значения, либо путем ввода, например:

```
{Присвоение переменной St1 строки символов}
St1 := 'Это символьная константа';
{Ввод строки символов St2 с клавиатуры}
Readln(St2);
```

Кроме этого применяется конструкция "переменная с индексом", что обеспечивает доступ к отдельным символам строки (таким же образом, как и к элементам массива типа ***Char***). Конкретный символ в строковой переменной обозначается с помощью ссылки на строковую переменную, за которой указывается индекс, определяющий позицию символа (например, ***St[5]***). При этом нижняя граница индекса равна ***1***. К элементу строки с индексом можно применить все операции, что и к переменной типа ***Char***.

Для строк определены операции присваивания, слияния (конкатенации) и сравнения.

Для сравнения строк применяются все операции отношения. Сравнение строк происходит посимвольно, начиная с первого символа. Строки равны, если имеют одинаковую длину и посимвольно эквивалентны.

Строки могут быть элементами списка ввода - вывода, при этом записывается имя строки без индекса. При вводе строковых переменных количество вводимых символов может быть меньше, чем длина строки. В этом случае вводимые символы размещаются с начала строки, а оставшиеся байты заполняются пробелами. Если количество вводимых символов превышает длину строки, лишние символы отбрасываются.

В *таблицах 5.4* и *5.5* приведены встроенные процедуры и функции Паскаля для работы со строками символов, а также примеры их использования.

Таблица 5.4 – Встроенные процедуры для работы со строками

Процедура	Описание	Примеры использования		
		Значение	Выражение	Результат
Delete(St, Poz, N)	Удаляет ***N*** символов из строки ***St***, начиная с позиции ***Poz***	***St = '1234567'***	***Delete(St, 4, 2)***	***St = '12367'***
Insert(St1, St2, Poz)	Добавляет в строку ***St2*** подстроку ***St1***, начиная с позиции ***Poz***	***St1 = 'аб'*** ***St2 = '1234'***	***Insert(St1, St2, 3)***	***St1 = 'аб'*** ***St2 = '12аб34'***
Str(X[:M[:N], St)	Преобразует вещественное ***X*** или целое ***I*** численное значение в его строковое представление ***St.*** ***M, N*** – необязательные параметры форматирования вывода (аналогичны форматам ***Write***)	***Целое*** ***I = 2567*** ***Веществ.*** ***X = 3.7e+03*** ***Веществ.*** ***X = 3.7e+03***	***Str(I:6, St)*** ***Str(X:10, St)*** ***Str(X:10:2, St)***	***St = '␣␣2567'*** ***St = '␣3.7e+0003'*** ***St = '␣␣3700.00'***
Val(St, X, Cod)	Преобразует строковое значение ***St*** в его численное представление ***X*** (вещественное или целое). В вспомогательной переменной ***Cod*** возвращается номер позиции первого ошибочного символа. ***Cod = 0***, если ошибок нет.	***St = '1234'*** ***St = '3.7e+03'*** ***St = '3.7+03'***	***Val(St, I, Cod)*** ***Val(St, X, Cod)*** ***Val(St, X, Cod)***	***Целое*** ***I = 1234*** ***Cod = 0*** ***Веществ.*** ***X = 3700.0*** ***Cod = 0*** ***Веществ.*** ***X = ?*** ***Cod = 4***

Таблица 5.5 – Встроенные функции для работы со строками

Функция	Описание	Примеры использования		
		Значение	Выражение	Результат
Length(St)	Возвращает динамическую длину .	***St = '1234␣abc'***	***L := Length(St)***	***L = 8***
Copy(St, Poz, N)	Выделяет из строки ***St*** подстроку длиной ***N*** символов, начиная с позиции ***Poz***	***St = '1234567'***	***S := Copy(St, 2, 4)***	***S = '2345'***

Функция	Описание	Примеры использования		
		Значение	Выражение	Результат
Pos(St1, St2)	Обнаруживает первое появление в строке ***St2*** подстроки ***St1***. Результат – номер той позиции, где находится первый символ подстроки ***St1***. Если в ***St2*** подстроки ***St1*** не найдено, результат равен нулю.	***St1 = 'гд'*** ***St2 = 'абвгдеж'*** ***St1 = '12'*** ***St2 = 'абвгдеж'***	***N := Pos(St1, St2)*** ***N := Pos(St1, St2)***	***N = 4*** ***N = 0***
UpCase(Ch)	Преобразует строчную латинскую букву ***Ch*** в заглавную.	***Ch = 'а'*** ***Ch = 'ы'***	***S := UpCase(Ch)*** ***S := UpCase(Ch)***	***S = 'А'*** ***S = 'ы'***
Concat(St1, St2, …, StN)	Выполняет конкатенацию (сцепление) последовательности строк ***St1, St2, …, StN***	***St1 = 'г.'*** ***St2 = '⌣⌣'*** ***St2 = 'Томск'***	***S := Concat(St1, St2, St3)***	***S = 'г.⌣⌣Томск'***
St1 + St2 + … + StN	Выполняет конкатенацию (сцепление) последовательности строк ***St1, St2, …, StN***	***St1 = 'г.'*** ***St2 = '⌣⌣'*** ***St2 = 'Томск'***	***S := St1 + St2 + St3***	***S = 'г.⌣⌣Томск'***

5.3 Множества

Множества представляют собой гибкий и наглядный механизм для решения многих задач. Например, их удобно использовать при обработке строк и текстов. В языке ***Borland Pascal*** множество означает то же, что и в математике.

> ***Множество*** - это ограниченная совокупность различных (неповторяющихся) элементов, имеющих общее свойство.

Тип элементов, составляющих конкретное множество (переменных или типизированных констант), называется ***базовым типом***. Максимальное количество значений базового типа множества называется его ***мощностью***.

> **Внимание!** В отличие от математики в ***Borland Pascal*** множества могут состоять только из элементов порядковых типов, мощность которых не превышает 256-ти значений[2]. Причем, порядковые значения верхней и нижней границы базового типа должны находиться в пределах диапазона от 0 до 255.

Множественный тип описывается с помощью служебных слов ***Set of***, например:

```
type
  tMn = Set of B;
```

Здесь ***tMn*** - множественный тип, ***B*** - базовый тип.

Пример описания переменной множественного типа:

---

[2] Вследствие этого в качестве базовых типов множеств не могут использоваться порядковые типы ***ShortInt, Integer, LongInt, Word***.

```
type
  tMnABCD = Set of 'A'..'D';
var
  MnABCD: tMnABCD;
```

Принадлежность переменных к множественному типу может быть определена и непосредственно в разделе описания переменных:

```
var
  Mn7: Set of 0..7;
```

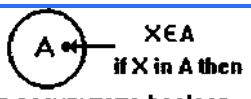
Константы множественного типа записываются в виде заключенной в квадратные скобки последовательности элементов или интервалов базового типа, разделенных запятыми, например:

['A', 'C'] [0, 2, 7] [3, 7, 11..14].

Константа вида ***[ ]*** означает пустое подмножество.

Операции, допустимые для работы с множествами с графическим пояснением приведены в *табл. 5.6*.

Таблица 5.6 - Операции работы с множествами

Математическое обозначение	Обозначение в Borland Pascal	Действие	Пример
∩	*	Пересечение	
∪	+	Объединение	
\	-	Разность	
∈	in	Принадлежность (элемента множеству)	
⊂ [⊆] ⊃ [⊇]	<= >=	Является подмножеством Включает подмножество	

В языке ***Borland Pascal*** используется операция ***in***, проверяющая принадлежность элемента базового типа, стоящего слева от знака операции, множеству, стоящему справа от знака операции. Результат выполнения этой операции - булевский. Операция проверки принадлежности элемента множеству часто используется вместо операций отношения, например:

'A' in ['A', 'B'] даст ***TRUE,***
2 in [1, 3, 6] даст ***FALSE.***

При использовании в программах данных множественного типа выполнение операций происходит над битовыми строками данных, причем каждому значению множественного типа в памяти компьютера соответствует один двоичный разряд.

6 Пример разработки алгоритма и программы обработки текстовых данных

6.1 Условия учебного задания

Определить количество слов в исходной строке текста. Слова могут отделяться друг от друга пробелами и (или) знаками препинания. Убрать повторяющиеся пробелы и знаки препинания и сформировать результирующую строку текста, в которой слова связаны символами подчеркивания.

6.2 Описание алгоритма

После циклического ввода строки исходного текста[3] выполняется контрольный вывод на экран введенных символов. Затем в теле внутреннего цикла осуществляют:

- уничтожение лишних пробелов и знаков препинания до начала первого слова,
- выделение слова,
- подсчет количества слов,
- накопление результирующей строки текста,
- удаление найденного слова.

Признак окончания внутреннего цикла – введена исходная строка без текста (просто нажат ***Enter***).

6.3 Листинг учебной программы

выше приведен листинг программы решения учебной задачи.

```
Program Primer8_1; {Демонстрация работы со строковыми переменными}
    {Условия задания}
  {Определить количество слов в исходной строке текста.
  Слова могут отделяться друг от друга пробелами и (или) знаками препинания.
  Убрать повторяющиеся пробелы и знаки препинания и сформировать результирующую
  строку текста, в которой слова связаны символами подчеркивания}

uses  {Подключение библиотек с подпрограммами и описаниями данных}
  Crt; {Подключаем модуль Паскаля с подпрограммами работы с экраном}

type {Объявление нестандартных типов данных}
  tRazd = set of char; {Создаем базовый тип данных - Множество символов}

const  {Объявление констант}
  {Типизированная константа - строка с тестовым примером}
  St : string =' г. Томск,  пр. Ленина, 40, ТУСУР, кафедра КИПР  ';

  {Типизированная константа, представляющая множество символов, разделяющих
  слова}
  Razd: tRazd = [' ', '.', ',', '-', ';', ':', '!', '?'];

Var {Инициализация переменных}
  Slovo,              {Строка для найденного слова}
  StRes: string;      {Результирующая строка}
  NumbSlovo,          {Счетчик слов}
  i,                  {Номер символа в строке}
  N: integer;     {Количество символов в слове}
```

[3] Цикл заканчивается, если введена «пустая» строка

```
Begin  {Начало основного блока программы Primer8_1}

 {Устанавливаем параметры экрана:}
 TextColor(Yellow);   { цвет символов желтый}
 TextBackGround(Blue);  { цвет фона синий}
 ClrScr;                          {Очистка экрана}

{Вывод назначения программы}
 Writeln('Демонстрация работы со строковыми переменными, Primer8_1');

    repeat {“Бесконечный” цикл ввода исходных данных}
       Writeln; {Пропуск строки}
       Writeln('Для завершения работы нажмите <<Enter>>');
       Write('Введите строку текста: ');
{Ввод исходной строки текста St при отладке отключен  с помощью фигурных скобок (как комментарий). В этом случае в программе используется значение типизированной константы с тестовым примером}
   {   ReadLn(St);   } {Временно блокируем ввод – переводим в комментарий}


       NumbSlovo := 0;        {Обнуляем счетчик слов}
       StRes := '';       {Подготавливаем начальное значение результирующей строки}

       if St = ''                    {Если введена пустая строка, }
          then Break; {то завершить цикл ввода и обработки строк}


       while St <> '' do          {Цикл поиска слов в исходной строке текста}
          begin
             WriteLn(#10#13'Исходная строка = <', St, '>'); Вспомог. печать}

      {Цикл удаления символов-разделителей слов  в начале строки}
       {Добиваемся, чтобы первый символ  строки был началом слова}
             while (St <> '') and (St[1] in Razd) do
                begin
                   WriteLn(' Удален лишний символ <', St[1], '>'); {Вспомог. печать}
                   Delete(St, 1, 1)
                end;

      {Возможен случай, когда слово в исходной строке единственное и не заканчивается символом-разделителем}
       Slovo := St;       {Копируем такую исходную строку в слово}
       N := Length(St); {Определяем длину слова}

    {Цикл выделения слова из исходной строки}
       for i := 1 to Length(St) do
        if (St <> '') and (St[i] in Razd)
          then begin {i-й символ исходной строки оказался разделителем слов }
                     N := i - 1; {Определяем длину слова}
                     Slovo := Copy(St, 1, N); {Копируем слово из исходной строки}
                     Break {слово найдено, завершаем цикл}
                end;

       if N = 0  {Если слово не выделено, выйти из цикла}
        then Break;
```

```
      NumbSlovo := NumbSlovo + 1; {Уточняем количество слов}
      Delete(St, 1, N); {Удаляем найденное слово из исходной строки}
      Writeln(' Из исходной строки удалено найденное слово <', Slovo, '>');

      {Формируем результирующую строку}
      {Если найденное слово не первое - добавляем в конец результирующей строки символ подчеркивания}
      if NumbSlovo > 1
          then StRes := StRes + '_';
      StRes := StRes + Slovo
  end;

    {Печать результатов}
    WriteLn(#10#13'   Результаты:');
    WriteLn('Найдено ', NumbSlovo, ' слов');
    WriteLn('Результирующая строка = <', StRes, '>');
    WriteLn('Для перехода ко вводу следующей строки нажмите <<Enter>>');
    Readln
  until false; {"бесконечный" цикл, выход – если введена пустая строка}

end. {Конец основного блока программы Primer8_1}
```

7 Индивидуальные задания

7.1 Требования к программе

В каждом варианте индивидуального задания исходной является строка с текстом. Слова в тексте разделяются пробелами и знаками препинания. Тестовый текст формируется самостоятельно. Он должен позволять оценить работоспособность программы. Целесообразно на период отладки тестовый пример оформить в виде тестовой константы.

При составлении программы по заданию предусмотреть:

- простейший диалог типа «запрос-ответ» при вводе данных;
- многократный ввод данных при исполнении программы, т.е. возможность повторной обработки при иных исходных данных. Признак окончания ввода данных – ввод пустой строки;
- вывод результатов в удобном для пользователя виде;
- освоить методику поиска причин и исправления ошибок, а также трассировки программы «по шагам»[4] по тестовому примеру.
- Предусмотреть вывод на экран *протокола тестирования*, в котором выводятся результаты преобразований на каждом этапе.

7.2 Варианты заданий

1. **Исключить из введенной строки комментарии, расположенные между парами скобок {} или (* *). Сами скобки также исключить.**
2. **Определить количество слов в строке текста (допустимые разделители пробелы и знак ;). Выяснить – является ли выделенное слово числом. Если да, то каким числом (целым или вещественным).**
3. **Найти во введенной строке текста некоторую последовательность символов и заменить ее иной последовательностью символов (замен может быть несколько).**

[4] Используйте функциональные клавиши F7 или F8.

4. Выделить в строке текста, состоящей только из одних цифр слова (допустимые разделители пробелы, знаки # и ;). Рассматривая каждое слово как число, определить сумму четных и нечетных чисел.

5. Выделить в строке текста, состоящей только из одних цифр и разделителей слова (допустимые разделители: пробелы, запятые, знаки $ и ;). Найти количество слов. Рассматривая каждое слово как число, найти минимальное и максимальное.

6. Дана строка, в которой слова произвольной длины состоят из нулей и единиц (разделители между словами: пробелы, запятые, знаки : и ;). Преобразовать эти двоичные коды в десятичную систему.

7. Выделить в строке символов, состоящей только из одних цифр и разделителей слова (допустимые разделители пробелы, запятые, знак ;). Рассматривая каждое слово как число, найти количество слов, делящихся на 3 без остатка

8. Выделить в строке текста, состоящей только из цифр, букв A, B, C, D, E, F слова, начинающиеся с # (окончание слова пробелы, запятые и знак ;). Рассматривая каждое такое слово как шестнадцатеричное число, перевести его в двоичную систему счисления.

9. Строка текста содержит информацию с фамилией (ФИО), годом и местом рождения. Сформировать и вывести на экран строку вида:

***Фамилия*: <ФИО>, *возраст:* <количество лет>, *родился в* <место рождения>**

10. Разбить текст, содержащийся в строке, на отдельные строки по 25 символов. Сформатировать каждую строку так, чтобы выровнять текст по правой границе путем вставки необходимого количества пробелов между словами.

11. Найти во введенной строке комментарии, расположенные между парами скобок {} или (* *) и отпечатать каждые из них с новой строки. Сами скобки не печатать. В строке произвольное количество комментариев.

12. Разработать программу шифровки и расшифровки строки символов. Алгоритм шифрования: буква А – заменяется на Я, буква Б на Ю и т.д. в обратном порядке алфавита. Предусмотреть, что символы в строке могут быть как на верхнем, так и на нижнем регистре.

13. Найти во введенной строке тексты, расположенные между ключевыми словами *begin* и *end* и отпечатать каждый из них с новой строки. В строке произвольное количество таких конструкций.

14. Разбить текст, содержащийся в строке, на отдельные строки по 30 символов. Сформатировать каждую строку так, чтобы «прижать» текст по правой границе путем вставки необходимого количества пробелов перед словами строки. Лишние пробелы между словами убрать.

15. Дана строка, в которой слова произвольной длины состоят из нулей и единиц (разделители между словами пробелы или запятые). Преобразовать эти двоичные слова в шестнадцатеричную систему.

16. Выделить в строке символов слова (допустимые разделители пробелы и запятые). Выяснить – является ли каждое из этих слов числом. Отсортировать найденные числа в порядке возрастания.

17. Определить, сколько раз встречается в строке каждый символ. Найти слово, содержащее максимальное количество самого распространенного символа.

18. Проверить, имеется ли в заданной строке символов баланс открывающих и закрывающих скобок. Учесть, что открывающая скобка всегда предшествует соответствующей закрывающей скобке. Предусмотреть также, что скобки могут быть вложенными.

19. В заданной строке найти последовательности символов, состоящие только из цифр. Преобразовать эти последовательности в числа и найти их среднее арифметическое.

20. Найти повторяющиеся слова (допустимые разделители: пробелы, запятые, знаки $ и ;) в двух строках. Определить, какое слово повторяется максимальное количество раз.

21. Удалить из строки текста все повторяющиеся слова и идущие подряд разделители (естественно, кроме первого). Допустимые разделители: пробелы, запятые, знаки ; ! и ?

22. Найти самое длинное общее слово в двух заданных строках текста. Допустимые разделители: пробелы, запятые, знаки ; ! и ?

23. Выделить в строке символов, состоящей только из цифр, букв A, B, C, D, E, F слова, начинающиеся с # (окончание слова пробелы, запятые, знаки : и ;). Рассматривая каждое такое слово как шестнадцатеричное число, перевести его в десятичную систему счисления.

24. В предложении все слова начинаются с различных букв. Вывести (если можно) слова предложения в таком порядке, чтобы последняя буква каждого слова совпадала с первой буквой следующего слова.

25. Даны два предложения. Найти самое короткое из слов первого предложения, которого нет во втором предложении.

Список рекомендуемой литературы

1. Основы информатики. Учеб. Пособие / Аладьев В.З., Хунт Ю.Я., Шишаков М.Л. - М.: Информационно-издательский дом "Филинъ", 1998. - 496 с.
2. Офицеров Д.В. и др. Программирование на персональных ЭВМ: Практикум: Учеб. пособие. –Мн.: Выш. Шк., 1993. – 256 с.
3. Марченко А.И., Марченко Л.А. Программирование в среде Borland Pascal 7.0. – К.: ЮНИОР, 1998. – 480 с.
4. Зуев Е.А. Программирование на языке Турбо Паскаль 6.0, 7.0. - М: Веста, Радио и связь, 1993. - 384 с.
5. Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0. - М.: "ДИАЛОГ-МИФИ", 1993. - 288 с.
6. Климова Л.М. Pascal 7.0: Практическое программирование. Решение типовых задач. –М.: КУДИЦ-ОБРАЗ, 2000. – 528 с.
7. Фаронов В.В. Turbo Pascal 7.0. Начальный курс. Учебное пособие.- М.: "НОЛИДЖ", 2001. - 576 с.
8. Фаронов В.В. Turbo Pascal 7.0. Практика программирования. Учебное пособие.- М.: "НОЛИДЖ", 1998. - 432 с.
9. ОС ТУСУР 6.1-97. Работы студенческие учебные и выпускные квалификационные. - Томск: ТУСУР, 1999.- 10 с.