Государственный комитет Российской Федерации
по высшему образованию

Самарский государственный аэрокосмический университет
имени академика С. П. Королева

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО СИМВОЛЬНОЙ ОБРАБОТКЕ В ТУРБО ПАСКАЛЕ

Методические указания к лабораторным работам

Самара 1996

Составители: Е. А. Логвинова, В. В. Семенов, Т.В.Макаренко

УДК 681.3.06

Лабораторный практикум по символьной обработке в Турбо Паскале: Метод. указания к лабораторным работам / Самар. гос. аэрокосм. ун-т ; Сост. Е. А. Логвинова, В. В. Семенов, Т. В. Макаренко. Самара, 1996. 12 с.

Содержатся рекомендации по выполнению лабораторных работ на популярном алгоритмическом языке программирования Турбо Паскаль для решения задач обработки символьных типов данных. Подробно рассматриваются стандартные процедуры и функции для обработки символьных переменных и строк. Приведены примеры программ и многочисленные варианты индивидуальных заданий к лабораторным работам по каждой теме. Предназначены для выполнения лабораторных работ по дисциплинам "ЭВМ и программирование", "Вычислительная техника и программирование", "Информатика" для студентов всех специальностей. Составлены на кафедре "Программное обеспечение вычислительных систем".

Печатаются по решению редакционно-издательского совета Самарского государственного аэрокосмического университета им. академика С. П. Королева

Рецензент А. О. Новиков

1. ОБРАБОТКА СИМВОЛЬНЫХ ТИПОВ ДАННЫХ

1.1. Символьные и строковые константы

Значением символьного (литерного) типа является множество всех символов компьютера. Это множество состоит из 256 различных символов, упорядоченных определенным образом, и содержит символы заглавных и строчных букв, цифр и других различных символов, включая специальные управляющие символы.

Каждому символу ставится в соответствие его код (целое число в диапазоне 0...255). Первая половина символов с кодами 0...127 соответствует стандарту ASCII (American Standard Code for Information Interchange - американский стандартный код для обмена информацией). Вторая половина символов с кодами 128...255 не ограничена жесткими рамками стандарта и может существенно различаться на разных компьютерах.

Символьная константа - это любой одиночный символ, заключенный в одинарные кавычки (апострофы). Для представления самого апострофа его изображение удваивается, например:

'*', 'a', 'X', ':', ''''.

Строковая константа - это последовательность любого, в том числе и равного нулю, количества символов, расположенных на одной строке и заключенных в апострофы. Если между апострофами нет ни одного символа, то такая строка называется нулевой строкой. Примеры именованных констант:

```
Const stroka = 'ПАСКАЛЬ';  { Строковая константа }
      symb   = 'П';        { Символьная константа }
```

1.2. Символьные и строковые переменные

Для описания символьных переменных используется тип **CHAR**, который относится к простым порядковым типам данных.

Для описания строковых переменных используется тип **STRING**. Описание строкового типа состоит из ключевого слова **String**, после которого

в квадратных скобках указывается максимальное количество символов строки данного типа (от 1 до 256). Если размер строки не указан, он считается равным 255. Строковую переменную можно рассматривать как одномерный массив символов (**ARRAY [0 .. N] Of Char**), однако в связи с широким использованием строк и некоторыми особенностями по сравнению со стандартными массивами они выделены в отдельный тип данных. Примеры описаний переменных символьного и строкового типов:

```
Var SYM  : Char;          { символьная переменная }
    STR  : String[50];    { строка ≤ 50 символов }
    MSTR: String;         { строка ≤ 255 символов }
```

В версии 7.0 для совместимости с другими языками программирования и средой Windows введен еще один вид символьной строки - строка, оканчивающаяся нулевым байтом - символом с кодом 0 (т. н. ASCIIZ - строка). В отличие от строк типа **STRING** у этих строк не накладывается ограничение на их размер (фактически размер может быть до 65535 байтов) Для этих строк введен новый стандартный тип **PCHAR**. Функции и процедуры для работы с ASCIIZ - строками содержатся в новом модуле **STRINGS**. Их рассмотрение выходит за рамки настоящих указаний, и мы отсылаем интересующихся к соответствующей литературе.

1.3. Стандартные функции и процедуры

Функция **ORD (C)** возвращает порядковый номер значения C любого порядкового типа. Применительно к символьному типу функция **ORD** дает код символа (положительное целое число в диапазоне от 0 до 255).

Например, после выполнения процедуры **WriteLn (Chr('Y'))** будет напечатано значение **89** (код символа **Y**).

Функция **CHR(B)** выполняет обратное преобразование, т. е. возвращает символ по его коду **B**.

В качестве примера приведем программу, которая выводит таблицу символов и соответствующих им кодов:

```
Program ASCII;
  Var i, j: Integer;
    Begin
      WriteLN ( ' Таблица символов:' );
      For i := 0 to 31 Do          { Номер строки }
        begin
        For j := 1 to 7 Do         { Номер столбца }
          begin
            Write( ' ', j*32+i:3, ' ', Chr( j*32+i ) );
          end;
        WriteLn('');
        end;
    End.
```

Функция **LENGTH (TXT)** возвращает целое значение, равное текущей длине строковой переменной **Txt** , т. е. значение 0 (если строка **Txt** не определена), или количество символов в ней после последней операции (не путать с максимальной длиной). Если **Txt** - символьная переменная или константа, то результат равен единице.

Процедура **STR (X, TXT)** преобразует число **X** любого вещественного или целого типов в строку символов **Txt** так, как это делает процедура **WriteLn** перед выводом. Можно также задать формат преобразования - ширину поля и количество символов в дробной части (для вещественного числа), указав соответствующие значения после **X** через двоеточия.

Процедура **VAL (TXT, X, CODE)** преобразует строковое выражение **Txt** во внутреннее представление целой или вещественной переменной **X**, которое определяется типом этой переменной, так как это делает процедура **ReadLn** Если преобразование прошло успешно, то параметр **Code** равен нулю, в противном случае он содержит номер позиции в строке **Txt**, где обнаружен ошибочный символ.

Функция **COPY (TXT, INDEX, NUM)** возвращает значение строкового типа, копируя из строки **Txt** фрагмент, начиная с символа с номером **Index** и длиной **Num**. Если **Index > Length (Txt)** , то результатом будет нулевая строка.

Функция **CONCAT (T1, T2, ... , Tn)** возвращает значение строкового типа, представляющее собой сцепление исходных строк **T1**, **T2**, ... , **Tn**. Суммарная длина исходных строковых выражений не должна превышать 255 символов.

Функция **POS (SUBST, TXT)** возвращает целое значение, равное наименьшему номеру позиции в строке **Txt,** с которой начинается идентичная **Subst** последовательность символов. Если искомая подстрока **Subst** в исходной строке **Txt** не найдена, то результат функции **POS** равен нулю.

Процедура **INSERT (SUBST, TXT, INDEX)** вставляет строковое выражение **Subst** в исходную строку **Txt,** начиная с символа с номером **Index**.

Процедура **DELETE (TXT, INDEX, NUM)** удаляет **Num** символов из строки **Txt**, начиная с символа с номером **Index**.

1.4. Операции над символьными и строковыми данными

Символьные и строковые переменные можно вводить с клавиатуры, присваивать им значения символьных выражений с помощью оператора присваивания (:=), объединять и сравнивать. Если вводимое или присваиваемое значение содержит больше символов, чем максимальная длина строковой переменной, то лишние символы отбрасываются.

Пример:

```
Type  STROKA = String [4];
Var   NName, LName: STROKA;
      Symb: Char;
  ...
      NName := 'Jan';      { Значение = 'Jan' , текущая длина = 3 }
      LName := 'Pascal';   { Значение = 'Pasc', текущая длина = 4 }
      Symb  := 'T';        { Значение = 'T' }
```

Сравнение строк происходит посимвольно, начиная с первых символов в строках. Строки равны, если имеют одинаковую длину и посимвольно эквивалентны (т. е. коды символов равны).

Примеры:

Логическое выражение:	Результат:
'A' > 'B'	FALSE
'jan' > 'Jan'	TRUE
'' = ''	TRUE
'Jan' = 'Jane'	FALSE
'2599' < '270'	TRUE

Объединение нескольких строк в одну может быть осуществлено с помощью оператора конкатенации (знак +) или функцией CONCAT, рассмотренной выше. Длина объединенной строки не должна превышать 255 символов.

Примеры:

Строковое выражение:	Результат:
'ada' + 'vla'	'adavla'
'5' + '.' + '4'	'5.4'

В качестве примера использования некоторых процедур и функций приведем фрагмент программы:

```
Var  Str1 : String [25];
     Str2 : String [10];
        I : Integer;
  ...
      Str1 := 'ФАНТАСТИКА';
      Delete ( Str1, 4, Length(Str1)-5 ); { Str1 = 'ФАНКА' }
      Str2 := Copy( Str1, 3, 2 ) + Str1;  { Str2 = 'НКФАНКА' }
      Insert ( 'abc', Str1, 5 );          { Str1 = 'ФАНКabcA' }
      I := Pos( Copy( Str1, 2, 3 ), Str2 );   { I = 4 }
```

1.5. Варианты заданий

1. Написать программу подсчета суммарного количества букв 'Л' и 'М' в данном тексте (произвольный текст вводится пользователем с клавиатуры).
2. Составить программу замены в произвольном тексте букв 'е' и 'Е' на 'р' и 'Р', а также наоборот. Например: **КЕРН ——> КРЕН**.
3. Написать программу, проверяющую, является ли частью данного произвольного текста слово '**милли**'. Ответ "Да" или "Нет".
4. Написать программу подсчета числа предложений в произвольном тексте (предложения разделяются точками, восклицательными и вопросительными знаками).
5. Написать программу подсчета числа слов в произвольном предложении. Слова разделяются пробелами, запятыми, двоеточиями и тире. Предлоги считаются словами.
6. Составить программу, которая определяет, имеется ли в произвольном тексте данное сочетание двух символов, например: ', -' .
7. Составить программу, удваивающую количество пробелов в произвольном тексте.
8. Составить программу нахождения самого короткого слова в произвольном тексте.
9. Составить программу подсчета общего количества цифр в произвольном тексте.
10. Составить программу, печатающую из произвольного текста слова, начинающиеся и кончающиеся на одну и ту же букву.
11. Составить программу, которая каждую встреченную в произвольном тексте букву 'Л' заменяет сочетанием букв 'ПА'. Например: **ЛУК ——> ПАУК**.
12. Составить программу, вычеркивающую из данного слова произвольную букву.
13. Составить программу, которая определяет, является ли данное слово "перевертышем". Например: **НАГАН, АННА**. Результатом работы программы должен быть ответ "Да" или "НЕТ".
14. Написать программу подсчета заданного слова в произвольном предложении. Предложение заканчивается точкой.
15. Написать программу, подсчитывающую, сколько раз в данном слове встречается сочетание 'АБ'. Например: **АБРАКАДАБРА**.
16. Написать программу, удваивающую каждую букву в данном слове.

17. Составить программу, подсчитывающую количество слов, совпадающих с последним словом.
18. Составить программу, которая выводит на печать все буквы, входящие в текст по одному разу.
19. Составить программу, которая разбивает предложение на слова и выводит их в столбец. Слова разделены пробелами, предложение заканчивается точкой.
20. Составить программу, которая распечатывает слова исходного предложения в обратном порядке. Слова разделяются пробелами, предложение заканчивается точкой. Например: **Я хочу домой.** ——> **Домой хочу я.**
21. Составить программу, которая переставляет первое слово в произвольном предложении в конец. Предложение заканчивается точкой.
22. Составить программу, подсчитывающую количество слов, начинающихся с того же символа, что и следующее слово.
23. Составить программу, подсчитывающую количество слов, совпадающих с первым словом.
24. Составить программу, исходными данными для которой служат две произвольные строки - **Str1** и **Str2**. Требуется определить номер позиции начала первого появления в **Str1** текста, содержащего символы **Str2** в обратном порядке. Например: **Str1** = '**внешность**' , **Str2** = '**сон**' . Номер позиции = 5, потому что строка '**нос**' (перевернутое - '**сон**') начинается в 5-й позиции слова '**внешность**'.
25. Написать программу, исходными данными для которой служат две произвольные строки - **Str1** и **Str2**. Требуется определить номер позиции начала последнего появления в **Str1** текста, содержащегося в **Str2**. Например: **Str1** = '**Миссисипи**' , **Str2** = '**си**'. Результат = 6.
26. Написать программу, которая печатает произвольный текст в обратном порядке. Например: **ОН ДАЛ** ——> **ЛАД НО** .
27. Написать программу, исходными данными для которой являются две произвольные текстовые строки - **Str1** и **Str2**, результатом работы - число, указывающее, сколько раз **Str2** встречается в **Str1**. **Str2** должна быть любой длины. Кроме того, любой символ в **Str1** может учитываться не более, чем в одном вхождении **Str2**. Например: **Str1** = '**балалайка**', **Str2** = '**ала**', результат = 1 (а не 2).
28. Написать программу, которая определяет в строке номер позиции первого символа, не являющегося буквой латинского алфавита.

29. Написать программу, которая определяет в произвольной строке номер позиции первой буквы, стоящей сразу после символа, не являющегося буквой. Например: **Stev9n** ——> Результат = 6.
30. Написать программу, которая из двух исходных строк **Str1** и **Str2** формирует новую строку **Rez**, в которой содержатся символы, присутствующие в обеих строках. Например: **Str1** = '**молоко**', **Str2** = '**кромка**', **Rez** = '**мок**'.

2. СТРОКОВЫЕ МАССИВЫ

2.1. Использование строковых массивов

Как указывалось выше, любую строковую переменную с максимальной длиной N можно рассматривать как одномерный массив из N+1 элементов типа **CHAR** (но не наоборот). Такой подход позволяет в некоторых случаях упростить программу, отказавшись от использования стандартных строковых подпрограмм.

Пример:

```
Var Name : String [8];
  . . .
    Name := 'Goldstar';
    Name [4] := 'F';     { Заменить четвертый символ на букву 'F' }
```

Менее изящный вариант замены в строке одного символа на другой с использованием стандартных процедур выглядит следующим образом:

```
    DELETE( Name, 4, 1 );   { Удалить четвертый символ из Name }
    INSERT ( 'F', Name, 4 ); { Вставить 'F' в четвертую позицию }
```

Следует помнить, что текущая длина строки хранится в нулевом символе, поэтому функция **Ord (Name[0])** даст такой же результат, что и функция **Length (Name)**.

В свою очередь символьные строки сами могут являться элементами одномерных и многомерных массивов. Их использование ничем не отличается от массивов, содержащих другие стандартные типы переменных.

Пример:

```
Var N: Array[1..2,4..6] Of String [12];   {Объявить массив N,содержащий }
                                          {2 строки и 3 столбца строк ≤ 12 }
  . . .
    N[1,5] := 'Gold';
    WriteLn ( N[1,5,3] );                 {Напечатать 3-й символ строки }
                                          {из 1-й строки 5-го столбца }
```

2.2. Варианты заданий

1. Дан одномерный массив текстовых строк. Написать программу, которая будет сортировать первые n элементов этого массива по возрастанию текущих длин.
2. Написать программу сортировки одномерного массива, состоящего из n строк в алфавитном порядке, основываясь на последнем символе каждой строки.
3. Написать программу, которая будет сортировать одномерный массив, состоящий из n строк в алфавитном порядке, основываясь на k-м символе каждой строки, где k является переменной, которую задает пользователь с клавиатуры. Если длина какой-то строки меньше k, то будем предполагать, что его k-м символом является пробел.
4. Написать программу, исходными данными для которой являются произвольный одномерный массив строк и целое число m. Задача состоит в том, чтобы преобразовать каждый элемент массива так, что каждая буква, входящая в [illegible]ое слово, была бы преобразована в новую букву, отстоящую от данной на m позиций "вниз" по алфавиту, все прочие символы (не относящиеся к буквам) сохраняются неизменными. Например, если значение параметра m=2, то буква 'a' должна быть преобразована в 'c'.
5. Написать программу шифровки произвольного латинского текста, заданного в виде одномерного массива строк с помощью "ключа" - символьной строки **Alpha**, которая представляет собой некоторую перестановку 26 букв латинского алфавита. Принцип преобразования состоит в следующем: если некоторая буква в шифруемой строке является k-й буквой в обычном алфавите, то вместо нее должна быть подставлена k-я буква "нового" алфавита **Alpha**. Например, если строка **Alpha** начинается с '**eps ...**', то все '**a**' должны быть заменены на '**e**', все '**b**' превратятся в '**p**' и т. д.
6. Написать программу расшифровки произвольного латинского текста, заданного в виде одномерного массива строк с помощью "ключа" - символьной строки **Alpha**, которая представляет собой некоторую перестановку 26 букв латинского алфавита. Принцип преобразования состоит в следующем: если некоторая буква в шифруемой строке является k-й буквой "нового" алфавита **Alpha**, то вместо нее должна быть подставлена k-я буква обычного алфавита. Например, если строка **Alpha** начинается с '**t x e ...**', то все '**t**' должны быть заменены на '**a**', все '**x**' превратятся в '**b**' и т. д.
7. Написать программу, которая преобразует значение календарной даты, заданной в форме 'ММ/ДД/ГГ (месяц-день-год), в форму 'МММ/ДД/ГГ'

(все даты относятся к XX веку). Например: дата 05.29.66 должна быть преобразована в 'Май 29, 1966'.

8. Написать программу аналогичную п.7, но с учетом того, что значения месяца и дня могут быть заданы одной цифрой (дата 05.03.66 может быть задана, как 5.3.66). Любое некорректное значение даты должно быть отвергнуто (соответствующее сообщение на экране и переход к повторному вводу данных).

9. Известно, что астрологи делят год на 12 периодов и каждому из них ставят в соответствие один из знаков Зодиака:

20.01÷18.02	— Водолей	23.07÷22.08	— Лев
19.02÷20.03	— Рыбы	23.08÷22.09	— Дева
21.03÷19.04	— Овен	23.09÷22.10	— Весы
20.04÷20.05	— Телец	23.10÷22.11	— Скорпион
21.05÷21.06	— Близнецы	23.11÷21.12	— Стрелец
22.06÷22.07	— Рак	22.12÷19.01	— Козерог

Написать программу, которая вводит дату некоторого дня года и печатает название соответствующего знака Зодиака.

10. Дан двумерный массив строк, содержащий фамилии и имена. Вывести список фамилий и соответствующих им имен в алфавитном порядке по первой букве фамилии.

РЕКОМЕНДУЕМЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0. М.: ДИАЛОГ - МИФИ, 1993. 288 с.

Пильщиков В. Н. Сборник упражнений по языку Паскаль: Учеб. пособие для вузов. М.: Наука: Гл. ред. физ. - мат. лит., 1989. 160 с.

Лабораторный практикум по символьной обработке в Турбо Паскале

Составители: Логвинова Елена Александровна
Семенов Валерий Владимирович
Макаренко Татьяна Васильевна

Редактор Т. И. Кузнецова

Техн. редактор Н. М. Каленюк

Подписано в печать 29.03.96 Формат 60 x 84 1/16
Бумага офсетная. Печать офсетная.
Усл.печ.л. 0,70. Уч.-изд.л. 0,8. Усл.кр.-отт. 0,93
Тираж 200 экз. Заказ 144

Самарский государственный аэрокосмический университет имени академика С. П. Королева.
443086 г. Самара, Московское шоссе, 34.

Издательство Самарского государственного аэрокосмического университета.
443001 г. Самара, ул. Ульяновская, 18.