Основы программирования

Учитель информатики и ИКТ
ГОУ г.Москвы СОШ №310
«У Чистых прудов»
Цыбикова Т.Р.

Тема 11.

ОБРАБОТКА СТРОК В ПАСКАЛЕ

СОДЕРЖАНИЕ

Символы в памяти компьютера

- В памяти компьютера могут храниться числа и символы.
- **Любой символ занимает один байт памяти**.
- Для данного, соответствующего одиночному символу, используется описатель **char**.
- Символы могут объединяться в массивы.
- Каждому **элементу** массива, как и числовому данному, соответствует **порядковый номер**, а **имя элемента** состоит из **имени всего массива и его собственного номера.**

Массив

- В тексте программы не всегда можно определить, какой массив обрабатывается: **числовой или символьный**, это можно понять только **по описанию массива.**
- Значение символьного данного — любой символ клавиатуры компьютера, ограниченный апострофами.
- *Например*: 'A', '?', '5' — значения символьных ве

```
var a: array [ 1.. 50 ] of char; x, y: char;
```

- Массив ***a*** может состоять из 50 символов, ему отводится при трансляции программы 50 байтов памяти.
- Элементы массива: a[1], a[2], ..., a[50]. Переменные ***x*** и ***y*** — простые, их значения одиночные символы.
- Для ввода символьного массива необходимо использовать цикл:

```
for i: = 1 to n do read (a[ i ]);
```

В содержание

Ввод массива из *n* символов

- При вводе такого массива достаточно набрать строку из ***n*** символов и в конце нажать <**Enter**>.
- Можно объявить в описании таблицу символов и для ее ввода использовать двойной цикл:

```
const n = 10; m = 15;
var b: array [ 1..n, 1:m ] of char; i, j: integer;
begin
   for i: =  1 to n do
      begin
         for j: =  1 to m do
            read (b [ i,j ]);
         writeln
      end;
end.
```

В примере используется **b** — таблица из 10 строк по 15 символов каждая. При ее вводе необходимо набирать строки по 15 символов и нажимать <Enter>. **Неудобство** такого ввода заключается в том, что все строки должны содержать по 15 символов, т. е. если набираются слова, то в них не может быть более чем 15 букв, а в коротких словах надо добавлять пробелы.

В содержание

ПРИМЕР.
Слово записать в обратном порядке

- При обработке символьных массивов используются такие же алгоритмы, как и для числовых.
- *Например*, **требуется слово**, заданное как массив символов, **записать в обратном порядке**, т. е. справа налево.
- При разработке алгоритма можно использовать такую постановку задачи:
  - данный числовой массив переписать так, чтобы последний элемент встал на первое место, предпоследний на второе и т. д., а первый — на последнее.
- Другими словами, необходимо из массива $a_1, a_2, \ldots, a_n$, получить $a_n, a_{n-1}, \ldots, a_1$, который будет находиться в массиве b (рис. 37).

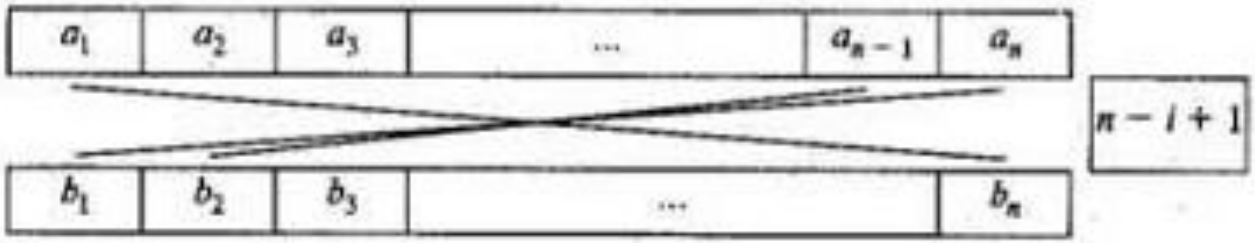

Рис. 37. Перемещение элементов из массива a в массив b и пересчет индексов

На *рисунке 37* в рамке обведена формула пересчета индекса:

когда у массива ***a*** номера перечисляются в прямом порядке, т. е. текущий индекс элемента массива изменяется ***от 1 до n***, у элементов массива ***b*** индексы должны меняться ***от n до 1.***

Такое изменение и обеспечивает данная формула, она приведена для индексов массива ***b***.

Программа E32 производит перемещение элементов в обратном порядке, для символьных данных она называется **программой обращения слова:**

```
program E28;
const n = 15;
var a, b: array [ 1.. n ] of char; i: integer;
begin
   for i: = 1 to n do
      begin
         read (a [ i ]);
         b [ n - i + 1 ]: = a [ i ]
      end;
   for i: = 1 to n do
      write (b [ i ])
end.
```

Строка

- Несколько подряд записанных символов образуют строку.
- **Строка — это ограниченная апострофами последовательность любых символов.**
- Длина строки, обрабатываемой в Паскале, не должна превышать 255 символов (апострофы не считаются).
- Это связано с тем, что в конце строки, в дополнительном байте, хранится её **длина** — количество символов, а наибольшее целое число, которое может быть записано в байте, — 255.
- Если требуется обработать текст, длина которого большее 255 знаков, то надо использовать массив строк.
- Описание строки имеет вид:

var x: string [20];

var x: string [20];

- Строка ***x*** должна быть не более чем из 20 символов.
- Если она меньше, то будет занимать в памяти столько байтов, сколько знаков она содержит (плюс 1 байт — длина).
- Поэтому при вводе строк нет необходимости дополнять их до указанной в описании длины.
- Для обработки строк используются **специальные операции** и собранные в специальную библиотеку **подпрограммы**.
- Операции позволяют работать со строками, как с цельными объектами, а подпрограммы, в основном, — с отдельными символами или частями строк.
- **Операции над строками — это объединение, сравнение и присваивание.**

Объединение строк

- **<u>Объединение строк.</u>**
- Эта операция позволяет объединить две строки в одну, присоединив начало второй строки к концу первой.
- Объединение обозначается знаком **«+»**.
- *Например*:

```
var x, y, z: string [ 10 ];
begin:
    x: = 'тепло';
    y: = 'ход';
    z: = x + y;
    write (z)
end.
```

a+b< >b+a

- Переменным ***x*** и ***y*** присваиваются значения строк, а переменной ***z*** — результат объединения этих строк в одну: 'теплоход'. При печати строки будет выдано содержимое области памяти, называемой ***z***.
- Очевидно, что операция объединения строк некоммутативная, т. е. для нее ***a+b<>b+a,*** поэтому при использовании объединения необходимо предусматривать, с какой стороны к данной строке присоединяется другая: **слева или справа**.
- Как и для арифметических операций, для данной операции со строками существует нейтральный элемент, не влияющий на ее результат. Это — строка нулевой длины (***пустая строка***), обозначаемая двумя рядом стоящими апострофами ('')'. Такую строку можно присоединить к любой строке слева или справа и строка не изменится.

Сравнение строк

- Для строк используются такие же операции отношения, как и для чисел, но они имеют несколько другой смысл.
- Если строки сравнивать на «равно» (=), то выполнение равенства означает **посимвольное совпадение строк**.
- Соответственно «не равно» (< >) означает **несовпадение хотя бы в одном знаке.**
- Остальные отношения (<, >) относятся к длинам строк, т. е. сравниваются не символы строк, а их количества.
- Если записать: *'a' < 'b' + 'c'*, то сначала выполнится **объединение строк** (эта операция имеет более высокий приоритет), а затем **сравнение длин**. В данном случае условие удовлетворяется, так как строка из одного символа меньше (по длине), чем строка из двух символов.

Присваивание

- Оператор присваивания для строковых данных имеет вид:

  Имя_строковой_переменной:= строковое выражение;
- Имя строковой переменной может быть простое или с индексом (элементом массива строк).
- Если в результате выполнения всех операций строкового выражения получается строка, длина которой превышает длину в описании переменной, стоящей слева от знака присваивания, то такая строка укорачивается справа до допустимой длины.

Пример

```
var x: string [ 6 ];
begin
    x: = 'мим' + 'озабоченный';
    write (x)
end.
```

- В результате работы этой программы будет напечатано слово «мимоза».
- Поэтому допустимая длина ***x*** — 6 символов, значение выражения справа от присваивания «мимозабоченный» сократится до «мимоза», остальные символы будут отброшены.

Длина строки

- . Функция длины строки выдает количество символов строки:

  length (строковое_выражение)

- Например:

```
program E29;
var x, y: string [ 20 ]; k, l, n: integer;
begin
  writeln ('введите две строки');
  readln (x); readln (y);
  k: = length (x); l: = length (y); n: = length (x + y);
  writeln ('длина первой строки': 25, 'длина второй строки': 25);
  writeln (k: 25, l: 25);
  writeln (x + y, 'длина строки', n)
end.
```

Программа E29

- В программе E29 используется **вывод с форматированием результата**.
- Первый раз формат (**:25**) указан подле строки, выводимой на экран (**'длина первой строки**').
- Это означает, что для данной строки отводится 25 позиций экрана, а поскольку выводимый текст короче (20 символов), он дополнится вначале пробелами, т. е. окажется правоуста- новленным в отведенном ему поле.
- Аналогично расположатся в предназначенном для них месте экрана целые числа — длины строк. Таким образом, результат работы программы будет иметь вид:

```
program E29;
var x, y: string [ 20 ]; k, l, n: integer;
begin
  writeln ('введите две строки');
  readln (x); readln (y);
  k: = length (x); l: = length (y); n: = length (x + y);
  writeln ('длина первой строки': 25, 'длина второй строки': 25);
  writeln (k: 25, l: 25);
  writeln (x + y, 'длина строки', n)
end.
```

```
длина первой строки        длина второй строки
                  7                         10
```

- С помощью форматирования можно располагать выводимые данные в столбцах, строить на экране дисплея таблицы.

Копирование строки или ее части

- Функция копирования называется также «вырезкой».
- Она позволяет скопировать одну область памяти в другую.
- Для копирования необходимо указать строковое выражение, из значения которого выделяется часть, а также начальный номер символа и количество символов копируемой части:

  copy (строковое_выражение, нач_номер_символа, кол-во_символов)
- *Например*, результатом работы функции **copy** ('информатика', 3, 5) будет слово 'форма'.

Разработка второй версии **программы обращения слова**

- Применим данную функцию для разработки второй версии **программы обращения слова.**
- Будем обрабатывать слово, выделяя из него буквы и присоединяя к результату слева.
- Переменной ***y***, содержащей результат, сначала присваивается значение пустой строки.
- Переменная цикла изменяет свои значения **от 1** (первого символа слова) **до длины вводимой строки** (номера последнего символа слова).

```
program E30;
var x, y: string [10 ]; i: integer;
begin
    write ('введите слово');
    readln (x);
    y: = ''; {присваивание результату начального значения —
    пустого слова}
    for i: =  1 to length (x) do
    y: =  copy (x, i, 1) + y; {присоединение копируемой буквы
    слева}
    writeln;
    write (y)
end.
```

Поиск подстроки в строке

- Функция поиска определяет, с какой позиции (номера символа) одна строка (подстрока) содержится в другой (данной строке).
- Если такое вхождение подстроки в строку имеет место, то результат работы функции — номер символа в исходной строке, с которого начинается подстрока.
- Если вхождения нет, то результат — нуль.
- Аргументы функции могут быть строковыми выражениями, **pos**(подстрока, исходная строка)

Вставка в строку

- В одну строку можно вставить другую строку, указав номер символа, начиная с которого осуществляется вставка.
- **Входные** данные процедуры — вставляемая строка, исходная строка и целочисленное выражение, задающее позицию вставки.
- Строки также могут быть заданы строковыми выражениями.
- Результат работы процедуры помещается в исходную строку, строка при этом «расширяется».
- Если длина вставки совместно с длиной исходной строки превышает допустимую длину исходной строки, то вставка укорачивается справа до допустимой длины.
- **Insert** (вставляемая строка, исходная строка, целочисленное выражение);

Удаление части строки

- Часть строки можно удалить, строка при этом «сжимается». Для удаления необходимо указать строку (в виде строкового выражения), начальный номер удаляемой части строки, количество удаляемых символов. Процедура удаления вызывается следующим образом:

  delete (строка, начальный номер, количество символов);

- Рассмотрим пример замены буквы в слове. Сделаем из слова «форма» слово «фирма».

```
program E31;
var x: string [10];
begin
   x: = 'форма';
   insert ('и', x, 2); {вставка буквы «и», получилось слово
   «фиорма»}
   delete (x, 3, 1); {удаление третьей буквы — буквы «о»}
   write (x)
end.
```

В содержание

Пример программы пословного перевода с английскою языка

- Пусть требуется построить программу-переводчик, которая бы, не учитывая правил грамматики, просто переводила каждое слово вводимого предложения.
- Поскольку программа демонстрационная, словари небольшие, содержат по 10 слов.
- Однако в случае расширения словарей программу можно использовать и для реального пословного перевода.

Идея выделения слов

- Идея выделения слов из вводимого предложения основана на том, что слова разделяются, как обычно, пробелами и слово между двумя пробелами вырезается.
- Далее происходит обращение к словарю и ищется совпадение выделенного слова со словами словаря.
- При обнаружении совпадения печатается слово из словаря переводов с тем же индексом элемента, как и в исходном словаре английских слов.
- Такая идея поиска может быть использована и в других таблицах, когда, например, по названию химического элемента ищется его масса.

Пример

- Рассмотрим этапы выполнения программы на примере перевода предложения «*I like a cat*.
- Для выделения слов из предложения будем использовать два указателя.
  - Первый из них — переменная ***m***, ее значение всегда ***1***, так как она указывает на первый символ вырезаемого (копируемого) из предложения слова.
  - Второй указатель — значение переменной ***k*** — всегда показывает на позицию пробела за выделяемым словом, являясь его номером.

Пример

- Для первого слова предложения первое значение переменной ***k*** — это ***2.***
- Функция поиска подстроки в строке работает таким образом, что поиск вхождения осуществляется обязательно с первой позиции исходной строки.
- Поэтому, если не менять исходную строку, то каждый раз найденным окажется первый пробел.
- *Следовательно,* после очередного выделения слова строку необходимо «усекать», оставляя только необработанную часть строки.

a:=copy (a, k, n-k+1);

- Такое «усечение» производит оператор:

  a:=copy (a, k, n-k+1);
- Здесь ***n-k+1*** длина оставшейся после выделения очередного слова части строки.
- Так, после выделения первого слова из предложения (слова «I»), которое будет присвоено переменной ***x***, строка ***a*** примет вид «like a cat».
- При поиске в словаре английских слов (массив ***e***) значение переменной ***x*** сравнивается с каждым словом словаря.

a:=copy (a, k, n-k+1);

- Если произошло совпадение, то печатается слово из словаря русских слов (массив ***r***) с таким же порядковым номером, как в английском словаре.
- Если весь словарь просмотрен, но слово не найдено, то печатается сообщение «слова в словаре нет». Программа обрабатывает определенный и неопределенный артикли, они в словаре не ищутся, происходит переход к обработке следующего слова предложения.

Программа имеет вид (начало):

```
program E32;
label 1, 2, 3;
const p = 10; {количество слов в каждом словаре}
var e, r: array [ 1..p ] of string [10 ]; x: string [ 10 ]; i, k, m, n:
integer; a, b: string[ 255 ];
begin
   {формирование словарей}
   writeln (' введите ', p, ' слов английских ');
   for i: = 1 to p do
      readln (e [ i ]);
   writeln ('введите русские слова ');
   for i: = 1 to p do
      readln (r [ i ]);
   {ввод и выделение слов из предложения}
   writeln (' введите предложение ');
   readln (b);
```

Программа имеет вид (продолжение):

```
    a: = b; {сохранение исходного предложения}
    m: = 1; n: = length (a); {n — длина предложения}
1: k: = pos (' ', a); {поиск пробела}
    if k = 0 then k: = n + 1; {выставление пробела после
    последнего слова}
    x: = copy (a, m, k – m); {вырезка слова длиной k – m}
    if (x = 'a') or (x = 'the') then goto 3; {обработка артиклей}
    {поиск слова в словаре}
    for i: = 1 to p do
        if e [ i ] = x then goto 2;
    writeln;
    writeln (' слова в словаре нет');
    goto 3;
2: writeln (r [ i ]); {выдача слова из русского словаря}
3: k: = k + 1; {переход к следующему слову в предложении}
    a: = copy (a, k, n – k + 1);
    if a <> ' ' then goto 1;
end.
```

Вопросы и задания

1. Чем отличается символьный тип данных от строковых?
2. Используя символьный массив, определите, сколько слов в данном тексте.
3. Используя символьный массив, посчитайте, сколько букв «а» в данном слове.
4. Используя средства обработки строк, исправьте слово «вылысыпыдысты».
5. Используя идею программы обращения слова E34, удвойте каждую букву в данном слове.
6. Используя программу обращения слова E34, определите, является ли данное слово палиндромом («перевертышем», например, «казак», «потоп», «кок» и т. д.).
7. Дана строка с несколькими запятыми. Подучите слово между первой и второй запятыми. Решите задачу с применением массива символов и строки символов.

Литература

- **А.А.Кузнецов, Н.В.Ипатова**
  «Основы информатики», 8-9 кл.:
  – Раздел 3. ОСНОВЫ ПРОГРАММИРОВАНИЯ, С.135-144