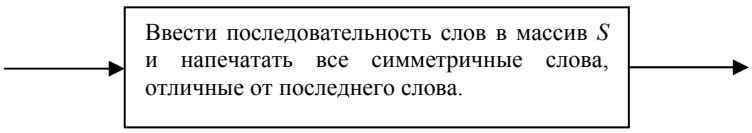
Вылиток А.А.

Структурное программирование и пошаговая детализация. Пример решения задачи

Задача. *Дана последовательность, содержащая от 2 до m слов (m≥2), в каждом из которых от 1 до n (n≥1) строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом – точка. Напечатать те слова последовательности, которые отличны от последнего слова и симметричны (т.е. читаются одинаково слева направо и справа налево).*
Построить структурированную схему и соответствующую этой схеме программу на языке Паскаль, используя метод пошаговой детализации.

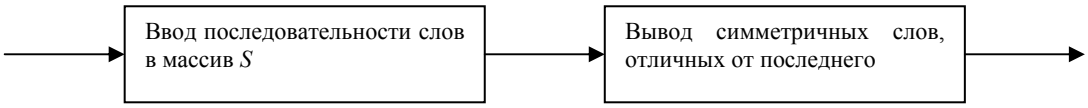
Решение

Для представления слова в программе будем использовать строку длины *n* (*слово=**packed array**[1..n] of char*). В случае необходимости введенное слово будем дополнять справа соответствующим числом пробелов. Поскольку каждое вводимое слово требуется сравнивать с последним словом, все вводимые слова следует сначала запомнить. Для этого воспользуемся массивом строк *S: **array**[1..m] of слово* .
Шаг1.



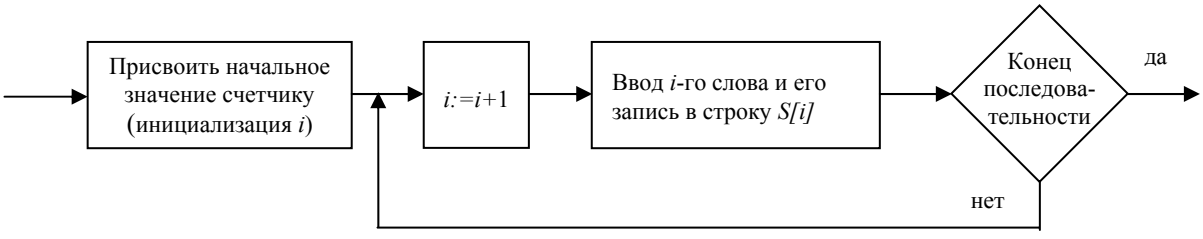
Шаг2.

Решение задачи очевидно разбивается на два последовательных блока: ввести данные в массив; затем, просматривая массив от начала до конца, напечатать те слова, которые удовлетворяют требуемым условиям.



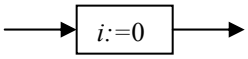
Шаг3а.

Детализируем сначала ввод последовательности слов: Очередное слово записывается в строку *S[i]*, где *i* – переменная-счетчик слов; счет начинается с 1, по окончании ввода значение *i* будет равно количеству введенных слов. Поскольку последовательность не пуста (есть хотя бы одно слово), организуем ввод, используя циклическую конструкцию с постусловием.

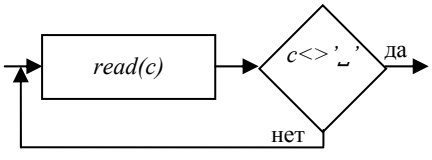


Шаг4а.

Теперь уточним блоки, входящие в конструкцию, полученную на шаге 3а. Для того, чтобы первое слово записывалось в *S[1]*, очевидно, начальное значение *i* должно быть нулевым.

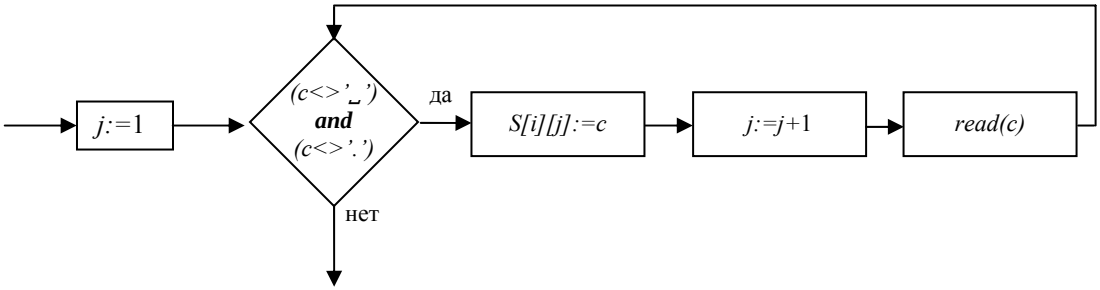


Для ввода слова понадобится вспомогательная символьная переменная *c: char*.
Ввод осуществляется посимвольно с помощью оператора *read(c)*.
Сначала нужно пропустить лишние пробелы:



Фрагмент программы, соответствующий этой блок-схеме, на Паскале записывается так:
repeat *read(c)* ***until*** *c< > '␣'* .

После пропуска пробелов вводим символы слова (по условию в слове есть хотя бы один символ). Для записи очередного введенного символа в *i*-е слово используем оператор *S[i][j]:=c*, где *j* – номер символа в слове, нумерация с единицы. Признак конца слова – очередной символ оказался пробелом или точкой. Пока не конец слова, записываем очередной символ в массив *S[i]* и считываем в переменную *c* следующий символ:



Соответствующий фрагмент на Паскале:
j:=1*;* ***while*** *(c<>'␣')* ***and*** *(c<>'.')* ***do begin*** *S[i][j]:=c; j:=j*+1*; read(c)* ***end***
Теперь запишем пробелы в оставшиеся позиции текущего слова *S[i]*. Пробелами нужно заполнить элементы массива *S[i]* с номерами от *j* до *n*. Приведем сразу фрагмент на Паскале:
for *j:=j* ***to*** *n* ***do*** *S[i][j]:= '␣'*
Если в *i*-м слове *n* символов (т.е. максимально возможное число символов в слове), то перед циклом ***for*** значение *j* будет равно n+1 и тело цикла ***for*** не выполнится ни разу.

Теперь мы можем вернуться к блок-схеме Шага 3а и записать соответствующий ей фрагмент на Паскале. Условие «Конец последовательности» в данной схеме можно теперь записать так: *c='.'*

```
{ввод последовательности слов в массив S:}
 i:=0;
repeat
  i:=i+1;
  { ввод i-го слова: }
   repeat  read(c)  until c< >'␣'; {пропустили пробелы; в c – первый символ i-го слова}
   j:=1; while (c<>'␣') and (c<>'.') do
            {записать очередной символ слова:}
           begin  S[i][j]:=c; j:=j+1; read(c) end ;
   {конец массива S[i] заполняем пробелами:}
    for j:=j  to n do  S[i][j]:= '␣'
 until   c='.'  { выходим из цикла, когда дошли до конца последовательности}
```

З А Д А Н И Е . Детализировать Шаг 3б: «вывод симметричных слов, отличных от последнего» (возможно, используя более мелкие шаги), построить окончательную блок-схему и, приведя необходимые описания типов и переменных, написать полную программу, решающую поставленную задачу.