# Лабораторная работа 16

#### 16.1 [#30]

Используя модуль Count3, напишите программу для подсчета реверсов в строке символов. Реверсом являются три последовательных символа, такие, что средний символ больше или меньше крайних.

Пример:

INPUT:1213453

OUTPUT:

Вход:1213453  
Количество реверсов:003

#### 16.2 [#30]

Проведите сборку и проверьте работу программы **TestRemove**. Тексты модуля и разделов проекта приведены ниже.   
В модуле восстановите процедуру AddQ.

**Текст модуля Count3**

UNIT Count3;

INTERFACE

VAR

Ones, Tens, Hundreds: CHAR;

PROCEDURE Start;

PROCEDURE Bump;

PROCEDURE Value (VAR V100, V10, V1: CHAR);

IMPLEMENTATION

PROCEDURE Start;

{Сбрасывает счетчик в ноль}

BEGIN{Start}

Ones := '0';

Tens := '0';

Hundreds := '0'

END;{Start}

PROCEDURE Bump;

{Увеличивает 3-цифровой счетчик определенный Ones, Tens, Hundreds

на единицу, если он находится в диапaзоне от 0 до 999 }

PROCEDURE NextDigit(VAR Digit: CHAR);

BEGIN {NextDigit}

IF Digit = '0' THEN Digit :='1' ELSE

IF Digit = '1' THEN Digit :='2' ELSE

IF Digit = '2' THEN Digit :='3' ELSE

IF Digit = '3' THEN Digit :='4' ELSE

IF Digit = '4' THEN Digit :='5' ELSE

IF Digit = '5' THEN Digit :='6' ELSE

IF Digit = '6' THEN Digit :='7' ELSE

IF Digit = '7' THEN Digit :='8' ELSE

IF Digit = '8' THEN Digit :='9' ELSE

IF Digit = '9' THEN Digit :='0'

END;{NextDigit}

BEGIN {Bump}

NextDigit( Ones );

IF Ones = '0'

THEN

BEGIN

NextDigit(Tens);

IF Tens= '0'

THEN

BEGIN

NextDigit(Hundreds);

IF Hundreds= '0'

THEN

BEGIN

Ones := '9';

Tens := '9';

Hundreds := '9'

END

END

END

END; {Bump}

PROCEDURE Value (VAR V100, V10, V1: CHAR);

{Возвращает содержимое счетчика}

BEGIN {Value}

V100 := Hundreds;

V10 := Tens;

V1 := Ones

END {Value};

BEGIN

END. {UNIT Count3}

Разделы проекта программы **TestRemove**

PROGRAM TestRemove(INPUT,OUTPUT);

{Читает строку из входа ,пропускает ее через RemoveExtraBlanks}

USES Queue;

VAR

Ch: CHAR;

PROCEDURE RemoveExtraBlanks;

{Удаляет лишниe пробелы между словами на одной строке}

BEGIN {TestRemove}

EmptyQ;

WRITE('Вход:');

WHILE NOT EOLN

DO

BEGIN

READ(Ch);

WRITE(Ch);

AddQ(Ch);

END;

WRITELN;

RemoveExtraBlanks;

WRITE('Выход:');

HeadQ(Ch);

WHILE Ch <> '#'

DO

BEGIN

WRITE(Ch);

DelQ;

HeadQ(Ch)

END;

WRITELN

END. {TestRemove}

**DP1**

PROCEDURE RemoveExtraBlanks;

{Удаляет лишниe пробелы между словами на одной строке}

VAR

Ch ,Blank, LineEnd: CHAR;

BEGIN {RemoveExtraBlanks}

Blank := ' ';

LineEnd :='$';

AddQ(LineEnd); {помечаем конец текста в очереди}

HeadQ(Ch);

{удаляем пробелы}

WHILE Ch <> LineEnd

DO

BEGIN

{читаем слово}

{удаляем пробелы}

{Вставляем пробел между словами}

END;

DelQ {удаяем LineEnd из очереди}

END; {RemoveExtraBlanks}

**DP1.1**

{удаляем пробелы}

WHILE Ch = Blank

DO

BEGIN

DelQ;

HeadQ(Ch)

END;

**DP1.2**

{читаем слово}

WHILE (Ch <> Blank) AND (Ch <> LineEnd)

DO

BEGIN

AddQ(Ch);

DelQ;

HeadQ(Ch)

END;

**DP1.3**

{Вставляем пробел между словами}

IF Ch <> LineEnd

THEN

AddQ(Blank)

**Модуль очереди**

UNIT Queue;

INTERFACE

PROCEDURE EmptyQ;

PROCEDURE AddQ (VAR Elt : CHAR);

PROCEDURE DelQ;

PROCEDURE HeadQ(VAR Elt: CHAR);

PROCEDURE WriteQ;

IMPLEMENTATION

VAR

Q, TEMP: TEXT;

PROCEDURE CopyOpen (VAR F1, F2 :TEXT);

{копирует строку из F1 в F2 без RESET или REWRITE;

таким образом F1 должен быть готов для чтения,а F2 для записи,

но прошлые строки у этих файлов могут быть не пусты }

VAR

Ch: CHAR;

BEGIN {CopyOpen}

WHILE NOT EOLN (F1)

DO

BEGIN

READ(F1,Ch);

WRITE(F2,Ch)

END

END;{CopyOpen}

PROCEDURE EmptyQ;

{Q := <,/,R>}

BEGIN {EmptyQ}

REWRITE(Q);

WRITELN(Q);

RESET(Q)

END{EmptyQ};

PROCEDURE AddQ (VAR Elt : CHAR);

{Q = <,x/,R>,где x строка И Elt = a -->

Q = <,xa/,R> }

BEGIN {AddQ}

.....

END {AddQ};

PROCEDURE DelQ;

{(Q = <,/,R> -->)|

(Q = <,ax/,R>,где a символ и x строка -->

Q:= <,x/,R> }

VAR

Ch: CHAR;

BEGIN {DelQ}

{удаляем первый элемент из Q};

READ(Q,Ch);

IF NOT EOF (Q)

THEN {не пустой}

BEGIN

REWRITE(Temp);

CopyOpen(Q,Temp);

WRITELN(Temp);

{копируем Temp в Q}

RESET(Temp);

REWRITE(Q);

CopyOpen(Temp,Q);

WRITELN(Q);

END;

RESET(Q)

END {DelQ};

PROCEDURE HeadQ(VAR Elt: CHAR);

{(Q = <,/,R> --> Elt := '#')|

(Q = <,ax/,R>,где a символ и x строка -->

Elt:= 'a' }

BEGIN {HeadQ}

IF NOT EOLN(Q)

THEN

READ(Q,Elt)

ELSE

Elt := '#';

RESET(Q)

END{HeadQ};

PROCEDURE WriteQ;

{ (Q = <,x/,R> и OUTPUT =<y,,W>,где y и x строка -->

OUTPUT := <y&x/,,W> }

BEGIN {WriteQ}

CopyOpen(Q,OUTPUT);

WRITELN(OUTPUT);

RESET(Q)

END{WriteQ};

BEGIN

END.