# Лабораторная работа 23

#### 23.1 [#20]

Из приведенных ниже разделов проекта соберите программу InsertionSort.

#### 23.2 [#20]

Для неотсортированной входной строки из 4 символов нарисуйте в тетради физическое и логическое представление данных на каждом шаге алгоритма сортировки и состояние переменных программы.

DP1

PROGRAM InsertionSort(INPUT, OUTPUT);

{Сортирует символы из INPUT}

CONST

Max = 16;

ListEnd = 0;

TYPE

RecArray = ARRAY [1 .. Max] OF

RECORD

Key: CHAR;

Next: 0 .. Max;

END;

VAR

Arr: RecArray;

First, Index: 0 .. Max;

Prev, Curr: 0 .. Max;

Extra: CHAR;

Found: BOOLEAN;

BEGIN {InsertionSort}

First := 0;

Index := 0;

WHILE NOT EOLN

DO

BEGIN

{Помещать запись в список, если позволяет пространство,

иначе игнорировать и сообщать об ошибке}

Index := Index + 1;

IF Index > Max

THEN

BEGIN

READ(Extra);

WRITELN('Сообщение содержит: ', Extra, '. Игнорируем.');

END

ELSE

BEGIN

READ(Arr[Index].Key);

{Включение Arr[Index] в связанный список}

END

END; {WHILE}

{Печать списка начиная с Arr[First]}

END. {InsertionSort}

DP 1.1

{Вставляем запись в связанный список}

Prev := 0;

Curr := First;

{Найти значения Prev и Curr, если существуют такие что

Arr[Prev].Key <= Arr[Index].Key <= Arr[Curr].Key}

Arr[Index].Next = Curr;

IF Prev = 0 {Первый элемент в списке}

THEN

First := Index;

ELSE

Arr[Prev].Next := Index;

DP 1.1.1

{Найти значения Prev и Curr, если существуют такие что

Arr[Prev].Key <= Arr[Index].Key <= Arr[Curr].Key}

Found := FALSE:

WHILE (Curr <> 0) AND NOT Found

DO

IF Arr[Index]. Key > Arr[Curr].Key

THEN

BEGIN

Prev := Curr;

Curr := Arr[Curr].Next

END

ELSE

Found := True;

DP 1.2

{Печать списка начиная с Arr[First]}

Index := First;

WHILE Index <> ListEnd

DO

BEGIN

WRITE(Arr[Index].Key);

Index := Arr[Index].Next

END;

WRITELN;

#### 23.3 [#20]

#### Соберите программу InsertSort2.

DP 2

PROGRAM InsertSort2 (INPUT, OUTPUT);

TYPE

NodePtr = ^Node;

Node = RECORD

Next: NodePtr;

Key: CHAR

END;

VAR

FirstPtr, NewPtr, Curr, Prev: NodePtr;

Found: BOOLEAN;

BEGIN {InsertSort2}

FirstPtr := NIL;

WHILE NOT EOLN

DO

BEGIN

NEW(NewPtr);

READ(NewPtr^.Key);

{2.1. Поместить NewPtr в надлежащее место}

END;

{2.2. Печать значений начиная с FirstPtr^.Key}

END. {InsertSort2}

DP 2.1

{2.1. Поместить NewPtr в надлежащее место}

Prev := NIL;

Curr := FirstPtr;

{2.1.1 Найдем значение Prev и Curr, такие что Prev^.Key <= NewPtr^.Key <= Curr^.Key}

NewPtr^.Next := Curr;

IF Prev = NIL

THEN

FirstPtr := NewPtr;

ELSE

Prev^.Next := NewPtr;

DP 2.1.1

{2.1.1 Найдем значение Prev и Curr, такие что Prev^.Key <= NewPtr^.Key <= Curr^.Key}

Found := FALSE;

WHILE (Curr <> NIL) AND NOT Found

DO

IF NewPtr^.Key > Curr^.Key

THEN

BEGIN

Prev := Curr;

Curr := Curr^.Next;

END

ELSE

Found := TRUE;

DP 2.2

{2.2. Печать значений начиная с FirstPtr^.Key}

NewPtr := FirstPtr;

WHILE NewPtr <> NIL

DO

BEGIN

WRITE(NewPtr^.Key);

NewPtr := NewPtr^.Next

END