# Лабораторная работа 23

#### 23.1 [#20]

#### Соберите программу InsertSort2.

#### 23.2 [#20]

Соберите программу TreeSort (в один этап). Для неотсортированной входной строки из 10 символов нарисуйте в тетради построенное программой бинарное дерево.

#### 23.3 [#30]

Сравните трудоемкость алгоритмов InsertSort2 и TreeSort в терминах количества просматриваемых пред вставкой элементов соответствующих связанных структур.

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

DP 3

PROGRAM TreeSort(INPUT, OUTPUT);

TYPE

Tree = ^NodeType;

NodeType = RECORD

Ch: CHAR

LLink, RLink: Tree;

END;

VAR

Root: Tree;

Ch: CHAR;

BEGIN {TreeSort}

Root := NIL;

WHILE NOT EOLN

DO

DEGIN

READ(Ch);

Insert(Root, Ch)

END;

PrintTree(Root)

END. {TreeSort}

DP 3.1

PROCEDURE Insert(Ptr:Tree, Data: CHAR);

BEGIN {Insert}

IF Ptr = NIL

THEN

BEGIN {Создаем лист со значением Data}

NEW(Ptr);

Ptr^.LLink := NIL;

Ptr^.RLink := NIL;

END

ELSE

IF Ptr^.Ch > Data

THEN

Insert(Ptr.LLink, Data)

ELSE

Insert(Ptr.RLink. Data)

END; {Insert}

DP 3.2

PROCEDURE PrintTree(Ptr: Tree);

BEGIN {PrintTree}

IF Ptr <> NIL

THEN {Печатает поддерево слева, вершину, поддерево справа}

BEGIN

PrintTree(Ptr^.LLink);

WRITE(Ch);

PrintTree(Ptr^.RLink);

END;

WRITELN

END; {PrintTree}