Type

BT=LongInt;   
U = ^BinTree;

BinTree = Record

Inf : BT;

L, R : U

End;

Покажем два варианта добавления элемента в дерево: итеративный и рекурсивный.

{Итеративный вариант добавления элемента в дерево, Turbo Pascal}

Procedure InsIteration(Var T : U; X : BT);

Var vsp, A : U;

Begin

New(A); A^.Inf := X; A^.L:=Nil; A^.R := Nil;

If T=Nil Then T:=A

Else Begin vsp := T;

While vsp <> Nil Do

If A^.Inf < vsp^.Inf

Then

If vsp^.L=Nil Then

Begin vsp^.L:=A; vsp:=A^.L End Else vsp:=vsp^.L

Else

If vsp^.R = Nil Then

Begin vsp^.R := A; vsp:=A^.R End Else vsp := vsp^.R;

End

End;

{Рекурсивный вариант добавления элемента в дерево, Turbo Pascal}

Procedure InsRec(Var Tree : U; x : BT);

Begin

If Tree = Nil

Then Begin

New(Tree);

Tree^.L := Nil;

Tree^.R := Nil;

Tree^.Inf := x

End

Else If x < Tree^.inf

Then InsRec(Tree^.L, x)

Else InsRec(Tree^.R, x)

End;

**program** exam20;

**type**

Tinf = **record**

name : string[10];

date : **array** [1..3] **of** integer;

price, count : integer;

**end**;

Tlist = ^RecList;

RecList = **record**

inf : Tinf;

next : Tlist;

**end**;

**procedure** Add\_Sort(**var** f : text; **var** Head, Tail : Tlist);

**var**

tmp, prev, walker : Tlist;

err : integer;

ch : char;

**procedure** pDate(**var** f : text; **var** point : Tlist);

**var**

i : integer;

saver : string[2];

**begin**

**for** i := 1 **to** 2 **do**

**begin**

read(f, saver);

val(saver, point^.inf.date[i], err);

**if** err<>0 **then**

**begin**

writeln('error ', err);

readln;

**end**;

read(f, ch);

**end**; **end**;

**begin**

**new**(tmp);

**while not** eof(f) **do**

**begin**

read(f, tmp^.inf.name);

pDate(f, tmp);

readln(f, tmp^.inf.date[3], tmp^.inf.price, tmp^.inf.count);

prev := head;

walker := prev^.next;

**while** (walker^.inf.name <= tmp^.inf.name) **and** (walker <> tail) **do**

**begin**

prev := walker;

walker := walker^.next;

**end**;

walker := prev;

prev := walker^.next;

**new**(walker^.next);

walker := walker^.next;

walker^.next := prev;

walker^.inf := tmp^.inf;

**end**; **end**;

**procedure** write\_stack(**var** f : text; head, tail : Tlist);

**var** p : Tlist; i : integer;

**begin**

p := Head^.next;

**while** (p <> tail) **do**

**begin**

write(f, p^.inf.name);

**for** i := 1 **to** 2 **do**

**begin**

write(f, p^.inf.date[i], '/');

**end**;

writeln(f, p^.inf.date[3], ' ', p^.inf.price, ' ', p^.inf.count);

p := p^.next;

**end**;

writeln(f);

**end**;

**procedure** Date\_sort(**var** Head, Tail : Tlist);

**var** p, tmp, prev, pred, trans : Tlist;

sp, st : real;

**begin**

pred := head;

p := head^.next;

**while** (p <> tail) **do**

**begin**

tmp := p;

sp := p^.inf.date[1] / 100 / 100;

sp := sp + p^.inf.date[2] / 100;

sp := sp + p^.inf.date[3];

**while** (tmp^.next <> tail) **do begin**

prev := tmp;

tmp := tmp^.next;

st := tmp^.inf.date[1] / 100 / 100;

st := st + tmp^.inf.date[2] / 100;

st := st + tmp^.inf.date[3];

**if** (sp > st) **then**

**begin**

**if** (p = prev) **then**

**begin**

pred^.next := tmp;

p^.next := tmp^.next;

tmp^.next := p;

**end**

**else begin**

pred^.next := tmp;

prev^.next := p;

trans := p^.next;

p^.next := tmp^.next;

tmp^.next := trans;

**end**;

trans := tmp;

tmp := p;

p := trans;

sp := st;

**end**;

**end**;

pred := p;

p := p^.next;

**end**;

**end**;

**var**

head, tail : Tlist;

f\_inp, f\_out : text;

**Begin**assign(f\_inp, 'input.txt');reset(f\_inp);

assign(f\_out, 'output.txt'); rewrite(f\_out);

**new**(head);

**new**(head^.next);

tail := head^.next;

Add\_Sort(f\_inp, head, tail);

write\_stack(f\_out, head, tail);

Date\_sort(head, tail);

write\_stack(f\_out, head, tail);

close(f\_inp); close(f\_out);

**end**.