**построить бинарное дерево и ввести N, определить количество узлов на N-ном уровне бинарного дерева**

program Project2;  
  
{$APPTYPE CONSOLE}  
  
uses  
SysUtils;  
  
type Ttree=^TSP;  
TSP=record  
key:integer;  
left, right:ttree;  
end;  
  
var tree:ttree; kol,n:integer; elem,i:integer; first:pointer;  
  
procedure insert(var x:TTree; y:integer);  
begin  
if x=nil then  
begin  
new(x);  
x^.key:=y;  
x^.left:=nil;  
x^.right:=nil;  
end  
else  
if y<=x^.key then insert(x^.left, y)  
else insert(x^.right, y);  
end;  
  
procedure obhod(x:ttree; level:integer);  
begin  
if x<>nil then  
begin  
if level=n then inc(kol);  
obhod(x^.left, level+1);  
obhod(x^.right, level+1);  
end;  
end;  
  
begin  
writeln('vvedite kol');  
readln(n);  
new(tree);  
first:=tree;  
writeln('write xhiclo');  
readln(tree^.key);  
tree^.left:=nil;  
tree^.right:=nil;  
for i:=1 to n-1 do  
begin  
writeln('vvedir chidlo');  
readln(elem);  
tree:=first;  
insert(tree, elem);  
end;  
writeln('write level');  
readln(n);  
kol:=0;  
obhod(tree, 1);  
writeln(kol);  
readln;  
end.

**условие Построить очередь, упорядочить её по убыванию, затем вставить эелемент к, не нарушив порядок очереди**

program r32423423423;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

PTSetOfTask = ^TSetOfTask;

TDataOfTask = record

id:integer;

end;

TSetOfTask = record

next: PTSetOfTask;

data: TDataOfTask;

end;

var

PFirstTask,Psecondtask,buf1,buf2: PTSetOfTask;

count:integer;

buf:TDataOfTask;

procedure AddInTasks(src: TDataOfTask; var base: PTSetOfTask); // neypor9d spisok

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := PFirsttask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInTask(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := Psecondtask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInSet2(src: TDataOfTask; var base: PTSetOfTask); // ypor9d sp, pervii pystoi

//is4em nyjnoe mesto dl9 vstavki

var

buf: PTSetOfTask;

p0, p: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.next := nil;

end;

new(p0);

p0^.next := nil;

p := base;

p0^.data := src;

while (p^.next <> nil) and (p0^.data.id<p^.next^.data.id) do

p := p^.next;

p0^.next := p^.next;

p^.next := p0;

end;

var

i:integer;

flag:boolean;

begin

writeln ('vvedite kol-vo v o4eredi') ;

readln(count);

//golovi 0

PFirstTask:=nil;

PSecondTask:=nil;

writeln ('vvedite vesa dl9 kajdogo') ;

//vvodim

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PFirstTask);

end;

buf1:=PFirstTask;

//vtoroi otsort sp

while (buf1<>nil) do

begin

addinset2(buf1.data,psecondtask);

buf1:=buf1^.next;

end;

writeln('o4ered` po ybivaniu');

//vivodim

buf1:=PSecondtask^.next;

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

writeln('dobavim v konec ne naryshiv por9dok');

readln(buf.id);

addinset2(buf,psecondtask);

writeln('vivod o4eredi posle dobavleni9 elemnta v konec');

//vivodim

buf1:=PSecondtask^.next;

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

readln;

end.

**есть построение дерева, вывод дерева, прямой обход, и поддерево с максимальным элементом**

**построить бинарное дерево поиска вывести элементы его поддерева на печать начиная с корня имеющего максимальный суммарный вес. Определить эту сумму и вывести на печать**

program Project2;

{$APPTYPE CONSOLE}

uses

SysUtils;

Type pt=^node;

node=record

data:integer; {Информационное поле}

left,right:pt; {Указатели на левого и правого потомков}

end;

Mas=array[1..100] of integer;

var y,i,n,sum1,sum2 :integer;

x: pt;

A: Mas;

str: string;

Procedure Insert(var x : pt; y:integer);{х - указатель на корень дерева, у - вставляемая вершина}

Begin

if x=nil then

Begin

new(x);

x^.data:=a[i];

x^.left:=nil;

x^.right:=nil;

write(str);

write (a[i]);

End

else if y<= x^.data then

begin

delete(str,1,4); insert (x^.left,y);

end

else

begin

str:=str+' ';insert(x^.right,y);

end;

End;

Procedure prym\_print(var x:pt;var sum:integer);

Begin

if x<> nil then

Begin

write ('(',x^.data,') ');

prym\_print(x^.left,sum);

sum:=sum+x^.data;

prym\_print(x^.right,sum);

End

else write ('');

End;

begin

x:= nil;

writeln ('vvedite kolichestvo elementov');

read (n);

writeln ('vvedite sami elementy');

for i:=1 to n do

read (A[i]);

write (str);

for i:=1 to n do begin

str:=(' ');

writeln (''); insert (x,a[i]); end; writeln('');

writeln ('pryamoj obhod dereva: ');

sum1:=0;

prym\_print (x,sum1);

sum2:=0;

prym\_print(x^.right,sum2);

writeln;

writeln(sum2);

readln;

readln;

end.

**задание: дан массив целых чисел. найти и распечатать все дубликаты в массиве,используя бинарное дерево**

program Laba1;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

ttree = ^ptree;

ptree = record

data : LongInt;

counter : longint;

left : ttree;

right : ttree;

end;

var

n, x, i : LongInt;

tree : ttree;

procedure add(var node : ttree; data : LongInt);

begin

if (node = nil) then

begin

New(node);

node^.data := data;

node^.counter := 1;

node^.left := nil;

node^.right := nil;

end

else

begin

if (data = node^.data) then inc(node^.counter)

else

if (data < node^.data) then add(node^.left, data)

else add(node^.right, data);

end;

end;

procedure printElement(data, count : LongInt);

var

i : LongInt;

begin

Writeln;

for i := 1 to count do write(data, ' ');

writeln;

end;

procedure TreeWalk(currentNode : ttree);

begin

if (currentNode = nil) then Exit;

if (currentNode^.counter <> 1) then printElement(currentNode^.data, currentNode^.counter);

TreeWalk(currentNode^.left);

TreeWalk(currentNode^.right);

end;

begin

writeln ('vvedite kolichestvo elementov masiva:');

readln(n);

writeln ('vvedite sami elementy');

for i := 1 to n do begin

readln(x);

Add(tree, x);

end;

TreeWalk(tree);

readln;

end.

**создать два однонаправленных списка затем из элементов обоих списков на овнове первого посторить список упорядоченный по возрастанию и распечатаеть его элементы**

program Project2;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

PTSetOfTask = ^TSetOfTask;

TDataOfTask = record

id:integer;

end;

TSetOfTask = record

next: PTSetOfTask;

data: TDataOfTask;

end;

var

PFirstTask,Psecondtask,buf1,PThirtTask: PTSetOfTask;

count:integer;

buf:TDataOfTask;

procedure AddInTasks(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := PFirsttask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInSet2(src: TDataOfTask; var base: PTSetOfTask); // первый элемент пустой для удобства.

//основан на том, что мы ищем нужное место вставки и таким образом получаем отстортированный массив

var

p0, p: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.next := nil;

end;

new(p0);

p0^.next := nil;

p := base;

p0^.data := src;

while (p^.next <> nil) and (p0^.data.id>p^.next^.data.id) do

p := p^.next;

p0^.next := p^.next;

p^.next := p0;

end;

var

i:integer;

begin

writeln ('Vvedite kolichestvo elementov v 1 spiske') ;

readln(count);

//обнуляем головы

PFirstTask:=nil;

PSecondTask:=nil;

PThirtTask:=nil;

//ввводим

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PFirstTask);

end;

writeln ('Vvedite kolichestvo elementov vo 2 spiske') ;

readln(count);

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PThirtTask);

end;

buf1:=PFirstTask;

//добавляем из 1-го списка

while (buf1<>nil) do

begin

addinset2(buf1.data,psecondtask);

buf1:=buf1^.next;

end;

//добавляем из 2-го списка

buf1:=PThirtTask;

while (buf1<>nil) do

begin

addinset2(buf1.data,psecondtask);

buf1:=buf1^.next;

end;

buf1:=PSecondtask^.next;

writeln('Vivod sort spiska');

//выводим

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

readln;

end.

**прямой обход, симметричный обход, проверка**

**есть дерево т по 2 потомка, прямой обход список преобразовать в списов=к симметричный обход**

program Preobraz;

{$APPTYPE CONSOLE}

uses

SysUtils;

Type pt=^node;

node=record

data:integer; {Информационное поле}

left,right:pt; {Указатели на левого и правого потомков}

end;

var m:array[1..8]of integer;

root :pt; {Указатель на корень дерева}

z,i,j,k:integer;

Procedure Insert(var x : pt; y:integer);//х - указатель на корень дерева, у -

Begin // вставляемая вершина}

if x=nil then

Begin

new(x);

x^.data:=y;

x^.left:=nil;

x^.right:=nil;

End

else if y<= x^.data then insert (x^.left,y)

else insert (x^.right,y);

End;

procedure printtree(temp:pt; h:integer);

var

i:integer;

begin

if temp<>nil

then

with temp^ do

begin

printtree(right,h+4);

for i:=1 to h do

write(' ');

writeln(data:30);

printtree(left,h+4)

end;

end;

Procedure prym\_print(var x:pt);

Begin

if x<> nil then

Begin

write (x^.data);

M[i]:=x^.data;

inc(i);

write(' ');

prym\_print(x^.left);

prym\_print(x^.right);

End;

End;

procedure preobrizpriamvsim();

begin

end;

Procedure sim\_print(var x:pt);

Begin

if x<> nil then

Begin

sim\_print(x^.left);

write (x^.data);

write(' ');

sim\_print(x^.right);

End;

end;

begin

for i:=1 to 7 do

begin

writeln('vvedite el');

readln(z);

insert(root,z);

end;

printtree(root,7);

i:=1;

writeln('priam\_obh');

prym\_print(root);

writeln;

for i := 1 to 8 do

for j := 1 to 8-i do

if M[j] > M[j+1] then begin

k := M[j];

M[j] := M[j+1];

M[j+1] := k

end;

write ('sim\_obhod:');

for i := 2 to 8 do

write (M[i]:4);

writeln;

writeln('proverka:');

sim\_print(root);

readln

end.

**создать два однонаправленных списка.затем оставить в первом списке только элементы,входящие в оба списка,и распечатать его элемент**

program Project17;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

PTSetOfTask = ^TSetOfTask;

TDataOfTask = record

id:integer;

end;

TSetOfTask = record

next: PTSetOfTask;

data: TDataOfTask;

end;

var

PFirstTask,Psecondtask,buf1,buf2: PTSetOfTask;

count:integer;

buf:TDataOfTask;

procedure AddInTasks(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := PFirsttask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInTask(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := Psecondtask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure DelTasks(var base: PTSetOfTask; id: integer);

var

buf, temp: PTSetOfTask;

begin

if (base^.data.id = id) then

begin

buf := base;

base := base^.next;

dispose(buf);

end

else

begin

buf := base;

while (buf^.next^.data.id <> id) do

begin

buf := buf^.next;

end;

temp := buf^.next;

buf^.next := buf^.next^.next;

buf := buf^.next;

dispose(temp);

end;

end;

var

i:integer;

flag:boolean;

begin

writeln ('kol-vo 1 m') ;

readln(count);

PFirstTask:=nil;

PSecondTask:=nil;

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PFirstTask);

end;

writeln ('kol-vo 2 m') ;

readln(count);

//Psecondtask

for i := 1 to count do

begin

readln(buf.id);

addintask(buf,PSecondTask);

end;

buf1:=PFirstTask;

while (buf1<>nil) do

begin

buf2:=PSecondtask;

flag:=false;

while (buf2<>nil) do

begin

if (buf2^.data.id=buf1^.data.id) then

begin

flag:=true;

end;

buf2:=buf2^.next;

end;

if (not flag) then

deltasks(pfirsttask,buf1^.data.id);

buf1:=buf1^.next;

end;

buf1:=PFirstTask;

writeln('vivod spiska posle ydaleni dyblikatov');

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

readln;

end.

Задание:

создать два однонаправленных списка затем из элементов обоих списков на овнове первого посторить список упорядоченный по возрастанию и распечатаеть его элементы

Код:

program Project17;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

PTSetOfTask = ^TSetOfTask;

TDataOfTask = record

id:integer;

end;

TSetOfTask = record

next: PTSetOfTask;

data: TDataOfTask;

end;

var

PFirstTask,Psecondtask,buf1,buf2,buf3,PThirtTask: PTSetOfTask;

count:integer;

buf:TDataOfTask;

{процедура формирования неупорядоченного списка}

procedure AddInTasks(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := PFirsttask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInSet2(src: TDataOfTask; var base: PTSetOfTask); // формируем упорядоченный, первый элемент пустой для удобства.

//основан на том, что мы ищем нужное место вставки и таким образом получаем отстортированный массив

var

buf: PTSetOfTask;

p0, p: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.next := nil;

end;

new(p0);

p0^.next := nil;

p := base;

p0^.data := src;

while (p^.next <> nil) and (p0^.data.id>p^.next^.data.id) do

p := p^.next;

p0^.next := p^.next;

p^.next := p0;

end;

var

i:integer;

flag:boolean;

begin

writeln ('Ввведите кол-вол элементов в 1-ом списке') ;

readln(count);

//обнуляем головы

PFirstTask:=nil;

PSecondTask:=nil;

PThirtTask:=nil;

//ввводим

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PFirstTask);

end;

writeln ('Ввведите кол-вол элементов в 2-ом списке') ;

readln(count);

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PThirtTask);

end;

buf1:=PFirstTask;

//добавляем из 1-го списка

while (buf1<>nil) do

begin

addinset2(buf1.data,psecondtask);

buf1:=buf1^.next;

end;

//добавляем из 2-го списка

buf1:=PThirtTask;

while (buf1<>nil) do

begin

addinset2(buf1.data,psecondtask);

buf1:=buf1^.next;

end;

buf1:=PSecondtask^.next;

writeln('Вывод сорт списка');

//выводим

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

readln;

end.

**Задание:**

**создать два однонаправленных списка.затем оставить в первом списке только элементы,входящие в оба списка,и распечатать его элемент**

код:

rogram Project17;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

PTSetOfTask = ^TSetOfTask;

TDataOfTask = record

id:integer;

end;

TSetOfTask = record

next: PTSetOfTask;

data: TDataOfTask;

end;

var

PFirstTask,Psecondtask,buf1,buf2: PTSetOfTask;

count:integer;

buf:TDataOfTask;

procedure AddInTasks(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := PFirsttask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInTask(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := Psecondtask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure DelTasks(var base: PTSetOfTask; id: integer);

var

buf, temp: PTSetOfTask;

begin

if (base^.data.id = id) then

begin

buf := base;

base := base^.next;

dispose(buf);

end

else

begin

buf := base;

while (buf^.next^.data.id <> id) do

begin

buf := buf^.next;

end;

temp := buf^.next;

buf^.next := buf^.next^.next;

buf := buf^.next;

dispose(temp);

end;

end;

var

i:integer;

flag:boolean;

begin

writeln ('Ввведите кол-вол элементов в массиве') ;

readln(count);

PFirstTask:=nil;

PSecondTask:=nil;

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PFirstTask);

end;

writeln ('Второй массив. Кол-во элементов?') ;

readln(count);

//Psecondtask

for i := 1 to count do

begin

readln(buf.id);

addintask(buf,PSecondTask);

end;

buf1:=PFirstTask;

while (buf1<>nil) do

begin

buf2:=PSecondtask;

flag:=false;

while (buf2<>nil) do

begin

if (buf2^.data.id=buf1^.data.id) then

begin

flag:=true;

end;

buf2:=buf2^.next;

end;

if (not flag) then

deltasks(pfirsttask,buf1^.data.id);

buf1:=buf1^.next;

end;

buf1:=PFirstTask;

writeln('Вывод списка 1 после удаления дубликатов');

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

readln;

end.

**Задание(даже не запускал, но идея такая. Хотя применять обход тут не есть хорошо):**

**вводим с клавиатуры массив из чисел. и все дубликаты, повторы выводим на экран**

**Код:**

program Project17;

{$APPTYPE CONSOLE}

uses

SysUtils;

TYPE

PTree = ^Tree;

Tree = record

data:Integer;

left,right:PTree;

end;

var

count,i,id:integer;

arrofelement:array of integer;

root:PTree;

flag:boolean;

function CheckElement:boolean;

begin

end;

procedure Add(var Element:PTree; inf:Integer);

begin

if Element=nil then

begin

New(Element);

Element^.data:=inf;

Element^.left:=nil;

Element^.right:=nil;

end

else

if inf<=Element^.data then Add(Element^.left,inf)

else Add(Element^.right,inf);

end;

Procedure prym\_print(t: ptree);

Begin

if t<> nil then

Begin

if (t^.data=id) then

begin

flag:=true;

exit;

end;

prym\_print(t^.left);

prym\_print(t^.right);

End;

End;

begin

writeln ('Ввведите кол-вол элементов в массиве') ;

readln(count);

setlength(arrofelement,count);

for i := 0 to count-1 do

begin

readln( arrofelement[i] );

add(root,arrofelement[i]);

end;

flag:=false;

for i := 0 to count-1 do

begin

prym\_print(prym\_print);

if (flag) then

writeln ('Дубликат ',arrofelement);

flag:=false;

end;

readln;

end.

**Задание (опять, даже не запускал)**

**найти поддерево с максимальным суммарным весом (считать и корень)**

**//рядом с exe должен быть txt c последовательностью для построения БД**

**код:**

program Project13;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

ss=^rec;

rec=record

data:integer;

ltag,rtag:boolean;

left,right:ss

end;

massive=array[1..1000] of integer;

var

mn:set of 1..100;

mns:set of 1..100;

mnss:set of 1..100;

number,local,z,buf,i:integer;

letter:char;

root,head,p,y:ss;

ArrOfUzl,arrofuzlobr:array of integer;

index,index1,max1,max2,sum:integer;

f:text;

procedure printtree(temp:ss; h:integer);

var

i:integer;

begin

if temp<>nil

then

with temp^ do

begin

printtree(right,h+4);

for i:=1 to h do

write(' ');

writeln(data:30);

printtree(left,h+4)

end;

end;

procedure directdetour(temp:ss);

begin

if temp = nil then write(0, ' ')

else begin

if temp^.data in mn then //write(temp^.data,' ')

else begin sum:=sum+temp^.data end;

directdetour(temp^.left);

write(temp^.data, ' ');

directdetour(temp^.right);

if temp^.data in mn then// write(temp^.data,' ')

else begin sum:=sum+temp^.data end;

end;

end;

procedure backdetour(temp:ss);

begin

if temp= nil then write(0,' ')

else

begin

write(temp^.data,' ');

backdetour(temp^.left);

write(temp^.data,' ');

backdetour(temp^.right);

if temp^.data in mns then write(temp^.data,' ')

else begin mns:=mns+[temp^.data]; write(temp^.data,'\* '); end;

end;

end;

procedure symmetricaldetour1(temp:ss);

begin

begin

if temp= nil then write(0,' ')

else

begin

write(temp^.data,' ');

symmetricaldetour1(temp^.left);

if temp^.data in mnss then write(temp^.data,' ')

else begin mnss:=mnss+[temp^.data]; write(temp^.data,'\* '); end;

write(temp^.data,' ');

symmetricaldetour1(temp^.right);

end;

end;

{ symmetricaldetour1(temp^.left);

write(' ',temp^.data,' ');

symmetricaldetour1(temp^.right);

end;}

end;

procedure symmetricaldetour(temp:ss);

procedure insertion(var p:ss);

begin

if y<>nil

then

if y^.right=nil

then

begin

y^.rtag:=false;

write(' ',y^.data,'->',p^.data,' ');

y^.right:=p

end

else

begin

y^.rtag:=true

end;

y:=p;

end;

begin

if temp<>nil

then

begin

symmetricaldetour(temp^.left);

insertion(temp);

symmetricaldetour(temp^.right);

end;

end;

procedure insert(var temp:ss; n:integer);

begin

if temp=nil

then

begin

new(temp);

with temp^ do

begin

data:=n;

left:=nil;

right:=nil;

end;

end

else

if n<temp^.data

then

insert(temp^.left,n)

else

insert(temp^.right,n);

end;

begin

assignfile(f,'tree.txt');

letter:='y';

root:=nil;

new(head);

head^.left:=root;

head^.right:=head;

p:=head^.left;

index:=0;

while letter='y' do

begin

reset(f);

writeln('enter the number of members:');

readln(number);

setlength(arrofuzl,number);

setlength(arrofuzlobr,number);

//arrofuzlobr

root:=nil;

y:=nil;

for z:=1 to number do

begin

read(f,local);

insert(root,local);

end;

writeln(' tree:');

printtree(root,0);

writeln('direct detour:');

directdetour(root);

writeln;

writeln('back detour:');

backdetour(root);

writeln;

index1:=0;

arrofuzlobr[number-1]:=arrofuzl[0];

max1:=0;

max2:=0;

sum:=0;

//анализируем левое поддерево

directdetour(root^.left);

max1:=sum+root^.data;

sum:=0;

directdetour(root^.right);

max2:=sum+root^.data;

if (max1>=max2) then

begin

writeln ('Сумма максимальная ',max1);

printtree(root^.left);

end

else

begin

writeln ('Сумма максимальная ',max1);

printtree(root^.right);

end;

writeln('continue? (Y/N)');

readln(letter);

close(f);

end;

readln

end.

**Задача://читерство**

**построить очередь упорядочить её по убыванию затем вставить в очередб эелемент k не нарушив порядок очереди**

12:39

Артём

КОд

program r32423423423;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

PTSetOfTask = ^TSetOfTask;

TDataOfTask = record

id:integer;

end;

TSetOfTask = record

next: PTSetOfTask;

data: TDataOfTask;

end;

var

PFirstTask,Psecondtask,buf1,buf2: PTSetOfTask;

count:integer;

buf:TDataOfTask;

procedure AddInTasks(src: TDataOfTask; var base: PTSetOfTask); // neypor9d spisok

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := PFirsttask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInTask(src: TDataOfTask; var base: PTSetOfTask);

var

buf: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.data := src;

base^.next := nil;

end

else

begin

buf := Psecondtask;

while (buf^.next <> nil) do

buf := buf^.next;

new(buf^.next);

buf := buf^.next;

buf^.data := src;

buf^.next := nil;

end;

end;

procedure AddInSet2(src: TDataOfTask; var base: PTSetOfTask); // ypor9d sp, pervii pystoi

//is4em nyjnoe mesto dl9 vstavki

var

buf: PTSetOfTask;

p0, p: PTSetOfTask;

begin

if (base = nil) then

begin

new(base);

base^.next := nil;

end;

new(p0);

p0^.next := nil;

p := base;

p0^.data := src;

while (p^.next <> nil) and (p0^.data.id<p^.next^.data.id) do

p := p^.next;

p0^.next := p^.next;

p^.next := p0;

end;

var

i:integer;

flag:boolean;

begin

writeln ('vvedite kol-vo v o4eredi') ;

readln(count);

//golovi 0

PFirstTask:=nil;

PSecondTask:=nil;

writeln ('vvedite vesa dl9 kajdogo') ;

//vvodim

for i := 1 to count do

begin

readln(buf.id);

addintasks(buf,PFirstTask);

end;

buf1:=PFirstTask;

//vtoroi otsort sp

while (buf1<>nil) do

begin

addinset2(buf1.data,psecondtask);

buf1:=buf1^.next;

end;

writeln('o4ered` po ybivaniu');

//vivodim

buf1:=PSecondtask^.next;

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

writeln('dobavim v konec ne naryshiv por9dok');

readln(buf.id);

addinset2(buf,psecondtask);

writeln('vivod o4eredi posle dobavleni9 elemnta v konec');

//vivodim

buf1:=PSecondtask^.next;

while (buf1<>nil) do

begin

write(buf1^.data.id:3);

buf1:=buf1^.next;

end;

readln;

end.

**Печать очереди, очередь удалена**

program queueMAz;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

ss=^rec;

rec=record

data:integer;

next:ss;

end;

var

start,finish:ss;

f:text;

local:integer;

procedure insert(var st,fin:ss);

var

temp:ss;

begin

new(temp);

temp^.data:=local;

temp^.next:=nil;

if st=nil

then

st:=temp

else

fin^.next:=temp;

fin:=temp;

end;

procedure delete;

var

temp1,temp2:ss;

begin

writeln('pechat ocherdi i udalenije:');

temp1:=start;

while temp1<>nil do

begin

temp2:=temp1;

write(temp1^.data,' ');

temp1:=temp1^.next;

dispose(temp2)

end;

writeln;

writeln('ochered udalena')

end;

begin

assignfile(f,'input.txt');

reset(f);

start:=nil;

finish:=nil;

while not eof(f) do

begin

read(f,local);

insert(start,finish);

end;

delete;

readln

end.  
  
 **список удаление всего списка и удаление элемента из списка**

program sp;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

ss=^spis;

spis=record

data:integer;

next:ss;

end;

var

head:ss;

f:text;

local:integer;

procedure insert;

var

temp1,temp2:ss;

begin

temp1:=head;

new(temp2);

temp2^.data:=local;

while (temp1^.next<>nil) and (temp1^.next^.data<local) do

temp1:=temp1^.next;

temp2^.next:=temp1^.next;

temp1^.next:=temp2;

end;

procedure print;

var

temp1:ss;

begin

temp1:=head^.next;

writeln('Spisok:');

while temp1<>nil do

begin

write(temp1^.data,' ');

temp1:=temp1^.next

end;

writeln;

end;

procedure delete;

var

temp1:ss;

begin

while head<>nil do

begin

temp1:=head;

head:=head^.next;

dispose(temp1);

end;

writeln('vse elemeni vmeste s golovnim udaleni')

end;

procedure deleteelement;

var

temp,temp1:ss;

a,k:integer;

flag:boolean;

begin

temp:=head;

flag:=false;

writeln('vvedite chislo');

readln(a);

while temp^.next<>nil do

begin

if temp^.next^.data=a

then

begin

flag:=true;

break

end;

temp:=temp^.next;

end;

if flag

then

begin

k:=temp^.next^.data;

temp1:=temp^.next;

temp^.next:=temp^.next^.next;

dispose(temp1);

writeln('udalen element ',k)

end;

end;

begin

assignfile(f,'input.txt');

reset(f);

new(head);

head^.next:=nil;

while not eof(f) do

begin

read(f,local);

insert;

end;

print;

deleteelement;

print;

delete;

readln

end.

**Печать и удаление стека полностью**

program stackqq;

{$APPTYPE CONSOLE}

uses

SysUtils;

type

ss=^rec;

rec=record

data:integer;

next:ss;

end;

var

stack:ss;

local:integer;

f:text;

procedure insert(var u:ss);

var

temp:ss;

begin

new(temp);

temp^.data:=local;

temp^.next:=u;

u:=temp;

end;

procedure delete;

var

temp1,temp2:ss;

begin

writeln('pechatr stecka i udalenije:');

temp1:=stack;

while temp1<>nil do

begin

temp2:=temp1;

write(temp1^.data,' ');

temp1:=temp1^.next;

dispose(temp2)

end;

writeln;

writeln('steck udalen')

end;

begin

assignfile(f,'input.txt');

reset(f);

stack:=nil;

while not eof(f) do

begin

read(f,local);

insert(stack);

end;

delete ;

readln

end.