**Двоичный поиск в массиве**

unit Unit2;

interface

const nok=0;

type

Tkey=integer;

Tinf=record

f:string;

k:integer;

end;

Tmas=array [1..50] of Tinf;

Tzad=class(Tobject)

a:Tmas;

n:Integer;

procedure sort;

function poiskd(k:Tkey):Tinf;

end;

implementation

procedure Tzad.sort;

var i,j:integer; z:Tinf;

begin

for i:=1 to n-1 do

for j:=i+1 to n do

if a[i].k>a[j].k then

begin

z:=a[i];

a[i]:=a[j];

a[j]:=z;

end;

end;

function Tzad.poiskd;

function del(L,R:word):integer;

var m:word;

begin

if r<=l then result:=r

else begin

m:=(l+r) div 2;

if a[m].k<k then del:=del(m+1,r)

else del:=del(l,m);

end;

end; //del

var i:integer;

begin

i:=del(1,n);

if a[i].k=k then result:=a[i]

else result.k:=nok;

end;

end.

**Сортировка слиянием массива**

unit Unit2;

interface

type

Tkey=integer;

Tinf=record

f:string;

k:Tkey;

end;

Tmas=array [1..50] of Tinf;

Tzad=class(Tobject)

a:Tmas;

n:Integer;

procedure sortslip;

end;

implementation

procedure Tzad.sortslip;

procedure slip(L,m,R:word);

var i,j,k:word; c:Tmas;

begin

i:=L;

j:=m+1;

k:=1;

while(i<=m) and (j<=r) do

if a[i].k<a[j].k then

begin

c[k]:=a[i];

inc(i);

inc(k);

end

else begin

c[k]:=a[j];

inc(k);

inc(j);

end;

while i<=m do

begin

c[k]:=a[i];

inc(i);

inc(k);

end;

while j<=r do

begin

c[k]:=a[j];

inc(k);

inc(j);

end;

k:=1;

for i:=l to r do

begin

a[i]:=c[k];

inc(k);

end;

end; //slip;

procedure srsl(L,R:word);

var m:Word;

begin

if l<>r then

begin

m:=(l+r) div 2;

srsl(l,m);

srsl(m+1,r);

slip(l,m,r);

end;

end;//srsl

begin

srsl(1,n);

end;

end.

**Квадарат вписанный в круг**

unit Unit2;

interface

uses Graphics;

type

tviz=class(Tobject)

x,y,r:Integer;

krcolor,kvcolor,bgcolor:Tcolor;

canvas:Tcanvas;

procedure ris; virtual;abstract;

procedure draw(bl:boolean);

procedure show;

procedure hide;

procedure moveto(dx,dy,dr:integer);

end;

tzad=class(Tviz)

constructor Create(x0,y0,r0:integer;canvas0:Tcanvas);

procedure ris; override;

end;

implementation

procedure Tviz.draw;

begin

if bl then

begin

krcolor:=clred;

kvcolor:=clgreen;

canvas.Pen.Color:=clblack;

end

else

begin

krcolor:=bgcolor;

kvcolor:=bgcolor;

canvas.Pen.Color:=bgcolor;

end;

ris;

end;

Procedure Tviz.show;

begin

draw(true);

end;

Procedure Tviz.hide;

begin

draw(False);

end;

procedure Tviz.moveto;

begin

hide;

x:=x+dx;

y:=y+dy;

r:=r+dr;

show;

end;

constructor Tzad.Create;

begin

inherited create;

x:=x0;

y:=y0;

r:=r0;

canvas:=canvas0;

bgcolor:=clwhite;

end;

procedure Tzad.ris;

var kvr:integer;

begin

canvas.Brush.Color:=krcolor;

canvas.Ellipse(x-r,y-r,x+r,y+r);

canvas.Brush.Color:=kvcolor;

kvr:=round(r/sqrt(2));

canvas.Rectangle(x-kvr,y-kvr,x+kvr,y+kvr);

end;

end.

**Полный перебор**

unit Unit2;

interface

Type

Tinf=record

w,c:integer;

end;

Tmas=array [1..50] of Tinf;

Tzad= class(Tobject)

a:Tmas;

wt,ct,wmax,cmax,n:Integer;

s,sopt:set of byte;

procedure PP(i:byte);

end;

implementation

procedure Tzad.PP;

begin

wt:=wt+a[i].w;

ct:=ct+a[i].c;

include(s,i);

if i<n then pp(i+1)

else if (wt<=wmax) and (ct>=cmax) then

begin

sopt:=s;

cmax:=ct;

end;

exclude(s,i);

wt:=wt-a[i].w;

ct:=ct-a[i].c;

if i<n then pp(i+1)

else if (wt<=wmax) and (ct>=cmax) then

begin

sopt:=s;

cmax:=ct;

end;end;end.

**Ветви и границы**

unit Unit2;

interface

type

Tinf=record

w,c:integer;

end;

Tmas=array [1..50] of Tinf;

Tzad=class(Tobject)

a:Tmas;

wmax,cmax,n:integer;

s,sopt:set of byte;

procedure VG(i,wt,oct:Integer);

end;

implementation

procedure Tzad.VG;

var wt1,oct1:Integer;

begin

//попытка включения

wt1:=wt+a[i].w;

if wt1<=wmax then

begin

include(s,i);

if i<n then vg(i+1,wt1,oct)

else if oct>=cmax then

begin

sopt:=s;

cmax:=oct;

end;

exclude(s,i);

end; //попытка исключения

oct1:=oct-a[i].c;

if oct1>cmax then

if i<n then vg(i+1,wt,oct1)

else begin

sopt:=s;

cmax:=oct1;

end;end;end.

**Сортировка хоара (quicksort)**

unit Unit2;

interface

type

tkey=integer;

Tinf=record

f:string;

k:Tkey;

end;

Tmas= array [1..50] of Tinf;

Tzad= class(Tobject)

a:Tmas;

n:Integer;

procedure quicksort;

end;

implementation

procedure Tzad.quicksort;

procedure sort(L,R:word);

var i,j:word; z:Tinf; x:Tkey;

begin

i:=L;

j:=R;

x:=a[(l+r)div 2].k;

repeat

while a[i].k<x do inc(i);

while a[j].k>x do dec(j);

if i<=j then

begin

z:=a[i];

a[i]:=a[j];

a[j]:=z;

inc(i);

dec(j);

end;

until j<i;

if L<=j then sort(L,j);

if i<=R then sort(i,R);

end;//sort

begin

sort(1,n);

end;end.

**сортировка очереди слиянием**

unit Unit2;

interface

type

Tkey=integer;

Tinf=record

f:string[50];

k:Tkey;

end;

Tself=^self;

self=record

inf:Tinf;

a:Tself;

end;

Tzad=class(Tobject)

sp1,sp,spk:Tself;

constructor Create;

procedure Addk(inf:Tinf);

procedure Read1(var inf:Tinf);

procedure sortslip;

end;

implementation

constructor Tzad.Create;

begin

inherited Create;

sp1:=nil;

spk:=nil;

end;

procedure Tzad.Addk;

begin

New(sp);

sp.inf:=inf;

sp.a:=nil;

if sp1=nil then

begin

sp1:=sp;

spk:=sp;

end

else begin

spk.a:=sp;

spk:=sp;

end;

end;

procedure Tzad.Read1;

begin

inf:=sp1.inf;

sp:=sp1;

sp1:=sp.a;

if sp1=nil then spk:=nil;

dispose(sp);

end;

procedure Tzad.sortslip;

procedure div2sp(tp:Tzad;var tq,tr:Tzad);

var c:boolean; inf:Tinf;

begin

tq:=Tzad.Create;

tr:=Tzad.Create;

c:=false;

while tp.sp1<>nil do

begin

c:=not(c);

tp.Read1(inf);

if c then tq.Addk(inf)

else tr.Addk(inf);

end;

end; //div2sp

procedure slip(tq,tr:Tzad; var tp:Tzad);

var inf:Tinf;

begin

while (tq.sp1<>nil)and(tr.sp1<>nil) do

if tq.sp1.inf.k<tr.sp1.inf.k then

begin

tq.Read1(inf);

tp.Addk(inf);

end

else begin

tr.Read1(inf);

tp.Addk(inf);

end;

while tq.sp1<>nil do

begin

tq.Read1(inf);

tp.Addk(inf);

end;

while tr.sp1<>nil do

begin

tr.Read1(inf);

tp.Addk(inf);

end;

end; //slip

procedure srsl(tp:Tzad);

var tq,tr:Tzad;

begin

if tp.sp1<>tp.spk then

begin

div2sp(tp,tq,tr);

srsl(tq);

srsl(tr);

slip(tq,tr,tp);

end;

end;//srsl

begin

srsl(self);

end;end.

**Сортировка стека с меткой пузырьком**

unit Unit2;

interface

type

Tkey=integer;

Tinf=record

f:String[50];

k:Tkey;

end;

Tself=^self;

self=record

inf:Tinf;

a:Tself;

end;

Tzad=class(Tobject)

sp,sp1:Tself;

constructor create;

procedure Add1(inf:Tinf);

procedure read1(var inf:Tinf);

procedure sortb;

end;

implementation

constructor Tzad.create;

begin

inherited create;

new(sp1);

sp1.a:=nil;

end;

procedure Tzad.Add1;

begin

new(sp);

sp.inf:=inf;

sp.a:=sp1.a;

sp1.a:=sp;

end;

procedure Tzad.read1;

begin

sp:=sp1.a;

inf:=sp.inf;

sp1.a:=sp.a;

dispose(sp);

end;

procedure Tzad.sortb;

procedure revafter(spi:Tself);

var sp:Tself;

begin

sp:=spi.a.a;

spi.a.a:=sp.a;

sp.a:=spi.a;

spi.a:=sp;

end;

var spt:Tself;

begin

spt:=nil;

repeat

sp:=sp1;

while sp.a.a<>spt do

begin

if sp.a.inf.k>sp.a.a.inf.k then revafter(sp);

sp:=sp.a;

end;

spt:=sp.a;

until spt=sp1.a.a;

end;end.

**Минимальный вес**

unit Unit2;

interface

type

tinf=record

w,c:integer;

end;

Tmas= array [1..20] of Tinf;

Tzad=class(Tobject)

a:Tmas;

n,wt,ct,cmax,wmax:integer;

sopt:set of byte;

procedure minw;

end;

implementation

procedure Tzad.minw;

function imin:integer;

var i,ic,wc:integer;

begin

ic:=0;

wc:=wmax;

for i:=1 to n do

if not(i in sopt) and (a[i].w<wc) then

begin

ic:=i;

wc:=a[i].w;

end;

result:=ic;

end; //imin

var i:integer;

begin

i:=imin;

wt:=a[i].w;

ct:=a[i].c;

while wt<=wmax do

begin

include(sopt,i);

i:=imin;

wt:=wt+a[i].w;

ct:=ct+a[i].c;

end;

cmax:=ct-a[i].c;

end;end.

**вычисление выражения в постфиксной форме**

unit Unit2;

interface

type

Tinf=char;

Tself=^self;

self=record

inf:Tinf;

a:Tself;

end;

TStack=class(Tobject)

sp1,sp:Tself;

constructor create;

procedure Add1(inf:Tinf);

procedure Read1(var inf:Tinf);

end;

Tzad=class(Tobject)

stack:Tstack;

zn:array ['a'..'я'] of extended;

function AV(stp:string):extended;

end;

implementation

constructor Tstack.create;

begin

inherited create;

sp1:=nil;

end;

procedure Tstack.Add1;

begin

new(sp);

sp.inf:=inf;

sp.a:=sp1;

sp1:=sp;

end;

procedure Tstack.Read1;

begin

sp:=sp1;

inf:=sp1.inf;

sp1:=sp1.a;

dispose(sp);

end;

function Tzad.AV;

var i:integer; chr,ch1,ch2,ch:char;

op1,op2,rez:extended;

begin

stack:=Tstack.create;

chr:=succ('z');

for i:=1 to length(stp) do

begin

ch:=stp[i];

if not(ch in ['+','-','\*','/','^']) then stack.Add1(ch)

else begin

stack.Read1(ch2);

stack.Read1(ch1);

op1:=zn[ch1];

op2:=zn[ch2];

case ch of

'+': rez:=op1+op2;

'-': rez:=op1-op2;

'\*': rez:=op1\*op2;

'/': rez:=op1/op2;

'^': rez:=exp(op2\*ln(op1));

end;

zn[chr]:=rez;

stack.Add1(chr);

inc(chr);

end;

end;

result:=rez;

stack.Free;

end;end.

**Перевод в постфиксную форму**

unit Unit2;

interface

type

Tinf=char;

Tself=^self;

self=record

inf:Tinf;

a:Tself;

end;

Tstack=class(Tobject)

sp1,sp:Tself;

constructor create;

procedure add1(inf:Tinf);

procedure read1(var inf:Tinf);

end;

Tzad=class(Tobject)

stack:Tstack;

procedure OBP(sti:string; var stp:string);

end;

implementation

constructor Tstack.create;

begin

inherited create;

sp1:=nil;

end;

procedure Tstack.add1;

begin

new(sp);

sp.inf:=inf;

sp.a:=sp1;

sp1:=sp;

end;

procedure Tstack.read1;

begin

inf:=sp1.inf;

sp:=sp1;

sp1:=sp1.a;

dispose(sp);

end;

procedure Tzad.OBP;

function prior(ch:char):byte;

begin

case ch of

'(',')': prior:=0;

'+','-': prior:=1;

'\*','/': prior:=2;

'^': prior:=3;

end;

end;//prior

var i:byte; pc:0..3; ch,ch1:char;

begin

stack:=Tstack.create;

stp:='';

for i:=1 to length(sti) do

begin

ch:=sti[i];

if not(ch in ['(',')','+','-','\*','/','^']) then

stp:=stp+ch

else

if ch='(' then stack.add1(ch)

else

if stack.sp1=nil then stack.add1(ch)

else

if ch=')' then

begin

stack.read1(ch);

while ch<>'(' do

begin

stp:=stp+ch;

stack.read1(ch);

end; end

else begin

pc:=prior(ch);

while(stack.sp1<>nil) and (pc<=prior(stack.sp1.inf)) do

begin

stack.read1(ch1);

stp:=stp+ch1;

end;

stack.add1(ch);

end;

end;

while stack.sp1<>nil do

begin

stack.read1(ch);

stp:=stp+ch;

end;

end;end.

**Ввод и вывод двоичного дерева поиска**

unit Unit2;

interface

uses sysutils, grids;

type

tkey=integer;

Tinf=record

f:string[50];

k:Tkey;

end;

Tself=^self;

self=record

inf:Tinf;

a1,a2:Tself;

end;

Tzad=class(Tobject)

p,proot,q,w:Tself;

constructor create;

procedure addb(inf:Tinf);

procedure wrt1b(var sgrid:Tstringgrid);

destructor free;

end;

implementation

constructor Tzad.create;

begin

inherited create;

proot:=nil;

end;

destructor Tzad.free;

procedure del(p:Tself);

begin

if p<>nil then

begin

del(p.a1);

del(p.a2);

dispose(p);

p:=nil;

end;

end;//del

begin

del(proot);

if self<>nil then

inherited destroy;

end;

procedure tzad.addb;

var bl:boolean;

begin

new(w);

w.inf:=inf;

w.a1:=nil;

w.a2:=nil;

if proot=nil then

proot:=w

else

begin

p:=proot;

repeat

q:=p;

bl:=inf.k<p.inf.k;

if bl then p:=p.a1

else p:=p.a2;

until p=nil;

if bl then q.a1:=w

else q.a2:=w;

end;

end;

procedure Tzad.wrt1b;

var i:word;

procedure wrt(p:Tself);

begin

if p<>nil then

begin

wrt(p.a1);

sgrid.Cells[0,i]:=p.inf.f;

sgrid.Cells[1,i]:=inttostr(p.inf.k);

inc(i);

wrt(p.a2);

end;

end;//wrt

begin

i:=1;

p:=proot;

wrt(p);end;end

**Хеш-таблица**

unit Unit2;

interface

const nok=0;

type

Tkey=integer;

tinf=record

f:string[50];

k:Tkey;

end;

Tself=^self;

self=record

inf:Tinf;

a:Tself;

end;

Tmas=array [1..1] of Tself;

Pmas=^Tmas;

Tzad=class(Tobject)

M:integer;

H:Pmas;

sp:Tself;

constructor create(M0:Integer);

procedure Add(inf:Tinf);

function Read(k:Tkey):Tinf;

end;

implementation

Constructor Tzad.create;

var i:word;

begin

inherited create;

m:=m0;

getmem(h,m\*4);

for i:=1 to m do

h[i]:=nil;

end;

procedure Tzad.Add;

var i:word;

begin

new(sp);

sp.inf:=inf;

i:=inf.k mod m;

sp.a:=h[i];

h[i]:=sp;

end;

function Tzad.Read;

var i:word;

begin

i:=k mod m;

sp:=h[i];

if sp<>nil then

begin

while (sp<>nil)and(sp.inf.k<>k) do

sp:=sp.a;

if sp.inf.k=k then result:=sp.inf

else result.k:=nok;

end

else result.k:=nok;

end;end.

**Двоичное сбалансированное дерево**

unit Unit2;

interface

type

Tkey=integer;

Tinf=record

f:string[50];

k:Tkey;

end;

Tmas=array [1..50] of tinf;

Tself=^self;

self=record

inf:Tinf;

a1,a2:Tself;

end;

Tzad=class(tobject)

p,proot:Tself;

constructor create;

Procedure blns(a:tmas;n:word);

function minkB:tinf;

destructor free;

end;

implementation

constructor Tzad.create;

begin

inherited create;

proot:=nil;

end;

Destructor Tzad.free;

procedure del(p:Tself);

begin

if p<>nil then

begin

del(p.a1);

del(p.a2);

dispose(p);

p:=nil;

end;

end;

begin

del(proot);

if self<>nil then

inherited destroy;

end;

Procedure Tzad.blns;

function bl(L,R:word):Tself;

var p:Tself; m:word;

begin

if r<l then p:=nil

else begin

m:=(l+r) div 2;

new(p);

p.inf:=a[m];

p.a1:=bl(L,m-1);

p.a2:=bl(m+1,R);

end;

result:=p;

end; //bl

begin

proot:=bl(1,n);

end;

function Tzad.minkB;

begin

p:=proot;

while p.a1<>nil do

p:=p.a1;

result:=p.inf;

end;end.

Двоичное дерево с поиском по ключу

unit Unit2;

interface

const nok=0;

type

Tkey=integer;

Tinf=record

f:string[50];

k:tkey;

end;

Tself=^self;

self=record

inf:Tinf;

a1,a2:Tself;

end;

Tzad=class(Tobject)

p,proot,q,w:Tself;

constructor create;

procedure addb(inf:Tinf);

function Pioskb(k:Tkey):Tinf;

destructor free;

end;

implementation

constructor Tzad.create;

begin

inherited create;

proot:=nil;

end;

destructor Tzad.free;

procedure del(p:Tself);

begin

if p<>nil then

begin

del(p.a1);

del(p.a2);

dispose(p);

p:=nil;

end;

end;

begin

del(proot);

if self<>nil then

inherited destroy;

end;

procedure Tzad.addb;

var bl:boolean;

begin

new(w);

w.inf:=inf;

w.a1:=nil;

w.a2:=nil;

if proot=nil then

proot:=w

else

begin

p:=proot;

repeat

q:=p;

bl:=inf.k<p.inf.k;

if bl then p:=p.a1

else p:=p.a2;

until p=nil;

if bl then q.a1:=w

else q.a2:=w;

end;end;

Function Tzad.Pioskb;

begin

p:=proot;

if p<>nil then

begin

while (p<>nil) and(p.inf.k<>k) do

if k<p.inf.k then p:=p.a1

else p:=p.a2;

if p<>nil then result:=p.inf

else result.k:=nok;

end

else result.k:=nok;

end;end.