```
/* Este algoritmo permite implementar un árbol B de orden n=2
Probar el algoritmo con por ejemplo los siguientes datos:


Insertar:10,25,7,30,8,15,40,5,42,30,32,46,13,22,18,35,26,38,24,45
,27

*/
#include<conio.h>
#include<stdio.h>
#include<alloc.h>
#define n 2
struct item{int kyy;
            int count;
            struct page *p;
            };
struct page{int m;
            struct page *p0;
            item e[2*n];  //n orden del árbol
            };

typedef struct page *puntero;
void mostrar(puntero p,int niv);

 // Función que permite insertar claves en el árbol B

void insertar(int x,puntero &a,int &h,item &v)
{int l,i,r;
puntero b;
item u;
if(a==NULL)   //carga el registro que contendrá la clave
{h=1;
  v.kyy=x;
  v.count=1;
  v.p=NULL;
  }
  else  // Busca donde insertar una nueva clave
  {l=1;
   r=(a->m);
    while (l<r)
    {i=(l+r)/2;
    if((a->e[i].kyy<=x))
      l=i+1;
    else
      r=i;
    }
   r--;
   if((r>=0)&&(a->e[r].kyy==x))   //repetidos
     {
     a->e[r].count++;
     h=0;
     }
```

```
   else{if ((r==0)&&(a->e[r].kyy>x)) // los mas chicos
         insertar(x,a->p0,h,u);
        else
         insertar(x,a->e[r].p,h,u); // los mayores que...
       if (h)
        if(a->m<2*n)
          {h=0;
           a->m++;
           for(int i=a->m-1;i>=(r+1);i--) // corrimiento para
insertar
              a->e[i]=a->e[i-1];
           if (a->e[r].kyy>x)// se ubica a el nuevo elemento
            a->e[r]=u;
           else
            a->e[r+1]=u;
           }
          else{//no hay lugar y se debe pedir una nueva página
               b=new(page);
               if(r<=n)
                {
                for(int i=0;i<n;i++)
                 b->e[i]=a->e[i+n];
                 if(r==n)
                 {
                  v=a->e[n];
                  b->e[0]=u;
                 }
                 else{
                       if (a->e[n-1].kyy < x)
                        v=u;
                       else
                       {
                        v=a->e[n-1];
                        if (a->e[0].kyy < x)
                          a->e[n-1]=u;
                        else
                         {
                         for(int i=n-1;i>=(r+1);i--)
                           a->e[i]=a->e[i-1];
                         a->e[r]=u; //+1
                         }
                       }
                      }
                 }
               else{
                  r=r-n;
                  v=a->e[n];
                  for(int i=0;i<r;i++)
                    b->e[i]=a->e[i+n+1];
                  b->e[r]=u;
                  for(int i=r+1;i<=n;i++)
                    b->e[i]=a->e[i+n];
```

```
                    }
             a->m=n; // se especifican los valores de m para las
páginas
             b->m=n;
             b->p0=v.p;// y los punteros correspondientes
             v.p=b;
             }
        }
        }
   }// fin inserción

 // Función que permite organizar las claves cuando las páginas
no cumplen con
//alguna de las características del árbol B.

void vacio(puntero &c,puntero &a,int &s,int &h)
{puntero b;
  int i,k,mb,mc;
  /*a pagina subocupada c pagina antecesora*/
  mc=c->m;
  if(s>mc)
  {/*b pagina a la derecha de a*/
   s++;
   b=c->e[s].p;
   mb=b->m;
   k=(mb-n+1)/2;/*numero de item de la pag ady b*/
   a->e[n]=c->e[s];
   a->e[n].p=b->p0;
    if (k>0)
    {/*mover k items de b a a */
     for(i=0;i<k-1;i++)
     a->e[i+n]=b->e[i];
     c->e[s]=b->e[k];
     c->e[s].p=b;
     b->p0=b->e[k].p;
     mb=mb-k;
     for(i=0;i<mb;i++)
     b->e[i]=b->e[i+k];
     b->m=mb;
     a->m=n-1+k;
     h=0;
     }
     else{/*unir pag a y b*/
          for(i=0;i<n;i++)
          a->e[i+n]=b->e[i];
          b=NULL;
          for(i=s;i<mc-1;i++)
          {
            c->e[i]=c->e[i+1];
            c->e[i].p=c->e[i+1].p;
          }
          a->m=2*n;
```

```
        c->m=mc-1;
        h=0;
        }
    }
else{/*b pagina a la izquierda de a*/
     if (s==0)
       b=c->p0;
     else
       b=c->e[s-1].p;
     mb=b->m;
     k=(mb-n+1)/2;
     if (k>0)
     {/*mover k items de la pagina b a la a */
      for(i=n-1;i>=0;i--)
        a->e[i+k]=a->e[i];
      a->e[k]=c->e[s];
      a->e[k].p=a->p0;
      mb=mb-k;
      for(i=k-1;i>=0;i++)
        a->e[i]=b->e[i+mb];
      a->p0=b->e[mb].p;
      c->e[s]=b->e[mb];
      c->e[s].p=a;
      b->m=mb-1;
      a->m=n-1+k;
      h=0;
      }

       else{/*unir pagina a co b */
           b->e[mb]=c->e[s];
           b->e[mb].p=a->p0;
           if (c->e[s].kyy==0)
             {
              puntero d=b->e[mb-1].p;
              puntero e=b->e[mb].p;
              for(i=0;i<e->m;i++)
                d->e[i+d->m]=e->e[i];
              d->m=d->m+e->m;
              b->e[mb-1].p=d;

              for(i=0;i<a->m;i++)
               { b->e[i+mb]=a->e[i];
                  b->e[i+mb].p=a->e[i].p;}
              b->m=mb=b->m+a->m;
              mostrar(b,0);
                }
           else{
             for(i=0;i<a->m;i++)
                b->e[i+mb+1]=a->e[i];
             mb=(b->m)+(a->m)+1;}
           if (mb<=2*n)
            { for(i=s ;i<c->m;i++)
```

```
                  c->e[i]=c->e[i+1];
                c->m=mc-1;
              }
             else  {
               i=(mb/2);
               c->e[c->m-1]=b->e[i];
               b->m=mb-i-1;
               for (int j=i+1;j<mb;j++)
                 a->e[j-(i+1)]=b->e[j];
               a->p0=b->e[i].p;
               a->m=i;
             // b->m=mb-1;
               c->e[c->m-1].p=a;
               c->p0=b;}
             h=c->m <=n;
             }
     }
     }


 // fin vacío


void sup(puntero &p,puntero &a,int &h,int &r)
{puntero q;
 int i;
q=p->e[p->m-1].p;
if(q!=NULL)

   {  sup(q,p,h,r);
   if (h==1)
    vacio(p,q,p->m-1,h);
  }
else

  {
  /* a->e[r-1]=p->e[0];
   for (i=1;i<p->m;i++)
    p->e[i-1]=p->e[i];
   a->e[r-1].p=p;
   p->m--;
   if (p->m<n)*/
     h=1;
   }
}

 // Función que permite suprimir claves del árbol B.

void suprimir(int x,puntero &a,int &h)
{
 int l,i=0,r;
 puntero q;
```

```
if(a==NULL)
{
 printf("\nNo esta el elemento   ");
 h=0;
 }
else
{
 l=1;
 r=a->m;/*Busqueda Binaria en el array*/
 while (l<r)
 {
  i=(l+r)/2;
  if (a->e[i].kyy<=x)
   l=i+1;
  else
   r=i;
  }

 if((r==1) &&(a->e[r-1].kyy>x))
    q=a->p0;
 else
  if((r==1) &&(a->e[r-1].kyy<x))
    q=a->e[r-1].p;
  else
    q=a->e[r-1].p;
 i=r-1;
 if((r<=a->m)&&(a->e[i].kyy==x))/*se encontro en e[i]*/
  {
    if (q==NULL)/*pagina terminal*/
     {
      for (int j=i;j<a->m;j++)
        a->e[j]=a->e[j+1];
      a->m--;
      if (a->m < n)
        h=1;
     }
    else
    {
    if (a->m==1)
    {
      a->e[i].kyy=0;
      h=1;
      }
     else
       sup(q,a,h,r);
     if(h==1)
        vacio(a,q,r-1,h);
        printf("%d",q->m);
    }
    }
 else
 {
```

```
   suprimir(x,q,h);
   if(h==1)
      vacio(a,q,r-1,h);
   }
}
}//fin suprimir

// Función que permite mostrar el Arbol B

void mostrar(puntero p,int niv)
{int i;
if(p!=NULL)
{
 for(i=0;i<niv;i++)
   printf("\n  ");
 for(i=0;i<p->m;i++)
   printf("%d  ",p->e[i].kyy);
 getchar();
 mostrar(p->p0,niv+1);
 for(i=0;i<p->m;i++)
   mostrar(p->e[i].p,niv+1);
 }
 }// fin mostrar

//*** PRINCIPAL ***

void main (void)
{
 puntero raiz,q;
 item v;
 int op,x,h;

 raiz=NULL;

 do
 {
   clrscr();
   printf("\n\n Menú \n");
   printf("\n 1_ Insertar\n");
   printf("\n 2_ Suprimir\n");
   printf("\n 3_ Mostrar\n");
   printf("\n 4_ Salir\n");
   printf("\n \n");

 printf("Ingrese opción  ");

 scanf("%d",&op);
 switch(op)
 {
 case 1:{
         printf("\n Ingrese clave a insertar(Finaliza con -1) ");
         scanf("%d",&x);
```

```
          while(x >=0)
          {
           insertar(x,raiz,h,v);
           if (h)
             {
              q=raiz;
              raiz=new(page);
              raiz->m=1;
              raiz->p0=q;
              raiz->e[0]=v;
              }
            printf("\n Ingrese clave a insertar (Finaliza con -1)
");
            scanf("%d",&x);
          }
          break;
          }
 case 2:{
           printf("ingrese clave a suprimir (Finaliza con -1) ");
           scanf("%d",&x);
           while(x >=0)
             {
               suprimir(x,raiz,h);
               if (h)
                 {
                   if (raiz->m == 0)
                     {
                      q=raiz;
                      raiz=q->p0;
                      printf("%d %d",raiz->m,q->p0->m);
                      getchar();
                      }
                  }
                 mostrar(raiz,0);
                 getchar();
                printf("ingrese clave a suprimir (Finaliza con -1)
");
                scanf("%d",&x);
               }
             break;
            }
 case 3:{
          mostrar(raiz,0);
          getchar();
           break;
          }
  }
}
while(op!=4);
}// fin principal
```