## *Segundo Parcial - Programación 2 - 05/03/2004*

a.- Se dispone de un árbol binario de búsqueda cuyos nodos contienen la siguiente información :

```
Clave:     Legajo (alfanumérica de 10 caracteres y/o números) .
Detalle:   fecha (que a su vez es una estructura)
```

Desarrolle una función de búsqueda que devuelva la dirección del nodo que contiene la clave buscada o `NULL` si no la encuentra .

Ingrese por teclado `Legajos` (Fin de datos = "`fin`"). Realice la búsqueda de cada clave en el árbol . Si la encuentra, mostrar la información de dicho nodo. De no encontrarla, mostrar mensaje aclaratorio.

**NOTA:** Asuma la existencia de una función que genera el árbol en memoria. Defina el prototipo de dicha función y úsela en consecuencia. (**No hay que desarrollarla)** .

b.- Escriba una clase Hora con los atributos que considere necesarios . Defina los tres constructores, operadores sobrecargados ++ (incrementa la hora en 1 segundo, en sus dos versiones), + (que sume segundos) y << . (ej: 23:58:00 + 122" --> 00:00:02)

c.- Declare una clase Punto con datos (x,y) ; la clase debe tener 3 constructores, una función que determine y devuelva la distancia al orígen de un objeto de la clase, funciones que obtengan los datos y operadores de extracción y de inserción .

Derive de Punto una clase Círculo, que tenga un dato "radio" y en el que las coordenadas de Punto representan el centro del círculo. Incluya constructores, operadores de extracción y de inserción y una función que determine si un objeto de la clase Punto está dentro del círculo, incluida la circunferencia .

Utilice accesos y tipo de herencia adecuados .

---

**No emplear ni suponer variables globales .**
**Las funciones deben recibir punteros (consulte) .**

***Parte a***

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

typedef struct
{
   int            di,
                  me,
                  an;
} t_fecha;

typedef struct
{
   char           Legajo[11];
   t_fecha        fecha;
} t_info;

typedef struct s_nodo
{
   t_info         info;
   struct s_nodo *izq,
                 *der;
} t_nodo;

t_nodo *busqueda(t_nodo *r, const char *Legajo)
{
   int            cmp;

      while(r && (cmp = strcmpi(Legajo, r->info.Legajo)) != 0)
      {
         if(cmp < 0)
            r = r->izq;
         else
            r = r->der;
      }

   return r;
}

/*****************************************************************************
 * estas funciones no se piden que esten desarrolladas, pero si sus prototipos
 *    y su uso desde main .
 */
int genera_arbol(t_nodo **r);
void vaciar_arbol(t_nodo **r);  /* P.S.:  esta no se pidio */

/*****************************************************************************
 *
 */
void main(void)
{
   t_nodo        *raiz,
                 *encontrado;
   char           Legajo[11];

   if(genera_arbol(&raiz))
   {
      printf("Legajo a buscar (para terminar ingrese  fin) : ");
      fflush(stdin);
      scanf("%s", Legajo);
/* o  gets(Legajo);
   o  fgets(Legajo, sizeof(Legajo), stdin);
      if(Legajo[strlen(Legajo) - 1] == '\n')
         Legajo[strlen(Legajo) - 1] = '\n';  no olvide matar el caracter  '\n' */
      while(strcmpi("fin", Legajo))
```

```
      {
         if((encontrado = busqueda(raiz, Legajo)) != NULL)
            printf("Encontrado - Fecha : %2d/%02d/%04d\n",
                   encontrado->info.fecha.di,
                   encontrado->info.fecha.me,
                   encontrado->info.fecha.an);
         else
            puts("*** No encontrado ***");
         printf("Legajo a buscar (para terminar ingrese  fin) : ");
         fflush(stdin);
         scanf("%s", Legajo);
      }
   }

   vaciar_arbol(&raiz);
} /* 128 lineas de facil y rapida escritura */

/*****************************************************************************
 * estas funciones no se piden que esten desarrolladas, pero si sus prototipos
 *    y su uso desde main .
 * para poder probar el programa, se hizo una funcion que carge informacion
 *    conocida de antemano, pudiendo ser modificada a gusto .
 */
int genera_arbol(t_nodo **r)
{                                        /* Legajo     di  me   an  */
   t_info         valores[]        = {{"b123",   { 10,  2, 2004 }},  /*raiz*/
                                      {"c432",   { 22,  3, 2000 }},  /*der */
                                      {"a-kxp1", {  5, 12, 1978 }}}; /*izq */
   int            posi             = 0;
   t_nodo       **ant;

   *r = NULL;
/* comentando desde aca  ... */
   while(posi < sizeof(valores) / sizeof(t_info))
   {
      ant = r;
      while(*ant)
         if(strcmp(valores[posi].Legajo, (*ant)->info.Legajo) < 0)
            ant = &(*ant)->izq;
         else
            ant = &(*ant)->der;
      if((*ant = (t_nodo *)malloc(sizeof(t_nodo))) == NULL)
         exit(puts("Memoria Insuficiente"));
      (*ant)->info = valores[posi];
      (*ant)->der = (*ant)->izq = NULL;
      printf("Cargando   : %-11s%2d/%02d/%04d\n",
             (*ant)->info.Legajo,
             (*ant)->info.fecha.di,
             (*ant)->info.fecha.me,
             (*ant)->info.fecha.an);
      posi++;
   }
/* ... hasta aca se puede probar con arbol vacio (con un warning por :  ant */

   return *r != NULL;
}

/*****************************************************************************
 *
 */
void vaciar_arbol(t_nodo **r)
{
   if(*r)
   {
      vaciar_arbol(&(*r)->izq);
      vaciar_arbol(&(*r)->der);
```

```
        printf("Eliminando : %-11s%2d/%02d/%04d\n",
                (*r)->info.Legajo,
                (*r)->info.fecha.di,
                (*r)->info.fecha.me,
                (*r)->info.fecha.an);
        *r = NULL;
    }
}
```

***Parte b***

```
#include <iostream.h>

class Hora
{
private :
   char           hh,
                  mm,
                  ss;
   void PonerEnHora(unsigned ho, unsigned mi, unsigned se);
public :
   Hora(void);
   Hora(unsigned ho, unsigned mi, unsigned se);
   Hora(const Hora &obj);
   Hora operator ++(void);
   Hora operator ++(int);
   Hora operator +(unsigned seg);
   friend ostream &operator <<(ostream &sal, const Hora &obj);
};

void Hora::PonerEnHora(unsigned ho, unsigned mi, unsigned se)
{
   ss = se % 60;
   mm = (mi + se / 60) % 60;
   hh = (ho + mi / 60 + se / 3600) % 24;
}

Hora::Hora(unsigned ho, unsigned mi, unsigned se)
{
   PonerEnHora(ho, mi, se);
}

Hora::Hora(void)
{
   PonerEnHora(0u, 0u, 0u);
}

Hora::Hora(const Hora &obj)
{
   PonerEnHora(obj.hh, obj.mm, obj.ss);
}

Hora Hora::operator ++(void)
{
   PonerEnHora(hh, mm, ss + 1);
   return *this;
}

Hora Hora::operator ++(int)
{
   Hora aux(*this);
   ++*this;
   return aux;
}

Hora Hora::operator +(unsigned seg)
{
   Hora aux;
   aux.PonerEnHora(hh, mm, ss + seg);
   return aux;
}

ostream &operator <<(ostream &sal, const Hora &obj)
{
   sal << int(obj.hh) << ':' << int(obj.mm) << ':' << int(obj.ss);
   return sal;
```

```
} /* apenas 66 lineas de codigo simple */

void main(void)
{
   Hora actual;
   cout << "actual contiene      " << actual << endl;
   cout << "actual++ da          " << actual++ << endl
        << " - y despues         " << actual << endl;
   cout << "++actual da          " << ++actual << endl;

   cout << "actual + 1d 1h 1m 1s " << actual + 90061 << endl;

   Hora otro(actual++);
   cout << "otro(actual++)       " << otro << endl
        << "actual               " << actual << endl;
}
```

### *Parte c*

```
#include <iostream.h>
#include <math.h>
class Punto
{
private :
   double   x,
            y;
public :
   Punto(void);
   Punto(double xx, double yy);
   Punto(const Punto &obj);
   double DistanciaAlOrigen(void);
   double ObtenerX(void);
   double ObtenerY(void);
   friend ostream &operator <<(ostream &sal, const Punto &obj);
   friend istream &operator >>(istream &ent, Punto &obj);
};

Punto::Punto(void) : x(0.0), y(0.0)
{
   /****/
}

Punto::Punto(double x, double y) : x(x), y(y)
{
   /****/
}

Punto::Punto(const Punto &obj) : x(obj.x), y(obj.y)
{
   /****/
}

double Punto::DistanciaAlOrigen(void)
{
   return sqrt(pow(x, 2.0) + pow(y, 2.0));
}

double Punto::ObtenerX(void)
{
   return x;
}

double Punto::ObtenerY(void)
{
   return y;
}

ostream &operator <<(ostream &sal, const Punto &obj)
{
   sal << '(' << obj.x << ", " << obj.y << ')';
   return sal;
}

istream &operator >>(istream &ent, Punto &obj)
{
   cout << "x = ";
   ent >> obj.x;
   cout << "y = ";
   ent >> obj.y;
   return ent;
}


class Circulo : public Punto
```

```
{
private :
   double   radio;

public :
   Circulo(void);
   Circulo(double x, double y, double radio);
   Circulo(const Circulo &obj);
   friend ostream &operator <<(ostream &sal, const Circulo &obj);
   friend istream &operator >>(istream &ent, Circulo &obj);
   bool IncluyeA(const Punto &obj);
};

Circulo::Circulo(void) : radio(0.0), Punto()
{
   /****/
}

Circulo::Circulo(double x, double y, double radio) : radio(radio), Punto(x, y)
{
   /****/
}

Circulo::Circulo(const Circulo &obj) : radio(obj.radio), Punto((Punto &)obj)
{
   /****/
}

ostream &operator <<(ostream &sal, const Circulo &obj)
{
   sal << "r = " << obj.radio << " centro = " << (Punto &)obj;
   return sal;
}
istream &operator >>(istream &ent, Circulo &obj)
{
   cout << "radio : ";
   ent >> obj.radio;
   cout << "coordenadas del centro" << endl;
   ent >> (Punto &)obj;
   return ent;
}

bool Circulo::IncluyeA(const Punto &obj)
{
   if(sqrt(pow(ObtenerX() - ((Punto &)obj).ObtenerX(), 2.0)  +
           pow(ObtenerY() - ((Punto &)obj).ObtenerY(), 2.0) <= radio))
      return true;
   return false;
} /* solo 115 lineas */

/*****************************************************************************
 *
 */
void main(void)
{
   Punto p1,
         p2(2, 4),
         p3(p2);

   cout << "p1             " << p1 << endl
        << "p2             " << p2 << endl
        << "p2             " << p3 << endl;
   cout << "cin >> p1      " << endl;
   cin >> p1;

   cout << "p1             " << p1 << endl
```

```
        << "(0, 0) - p1    " << p1.DistanciaAlOrigen() << endl;

   Circulo  c1,
            c2(-1, -1, 1),
            c3(c2);
   cout << "c1             " << c1 << endl
        << "c2             " << c2 << endl
        << "c3             " << c3 << endl;

   cout << "c3 incluye a p1 " << (c3.IncluyeA(p1) ? "SI" : "NO") << endl;

   char a[2]; cin.ignore(500, '\n'); cin.getline(a, sizeof(a), '\n');
}
```