PRÁCTICA 4

FUNDAMENTOS DE PROGRAMACIÓN

CURSO 2016-2017

INDICE

DATOS DEL ALUMNO:

NOMBRE: Javier

PRIMER APELLIDO : Navarrete

SEGUNDO APELLIDO: Rodoriguez

DNI: 46893078-b

E-MAIL: javiernavarreterodriguez@hotmail.com

ESTRUCTURAS DE DATOS:

Introducción General.

Para la realización de la practica se han utilizado tres TAD.

tad_datos_generales.

En este Tad se almacenan los datos generales tales como fecha actual, meses permitidos, etc.

CalendarioMes.

Aquí se define una arreglo de [numero_meses] de matrices de 31 filas (numero máximo de días mes) con columnas [horas días+2] de datos enumerados.

GestionReservasSala.

Consta de un vector de [numero_meses] donde se almacenan el numero de reservas/mes y cuatro listas con cinco datos de tipo int y un string para almacenar las reservas de cada mes (una lista por mes).

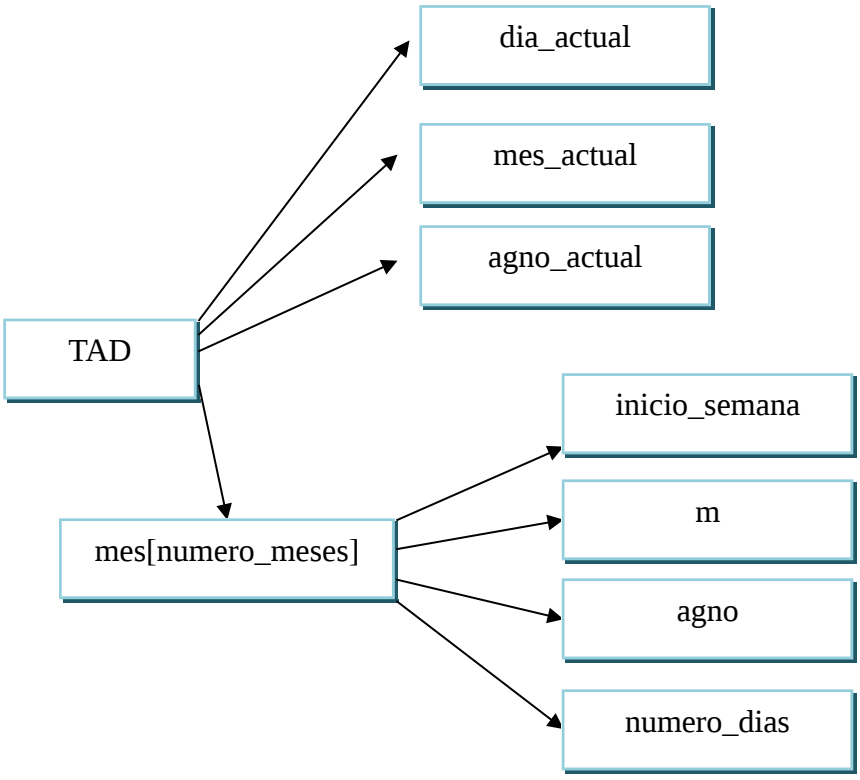
tad_datos_iniciales

Posee tres datos tipo int

dia_actual => Almacena el día local.
mes_actual=> Almacena el mes actual
agno_actual => Almacena el año actual.

Un vector de longitud [numero_meses] de estructuras (una para cada mes valido) de cuatro datos tipo int

inicio_semana => Almacena el día de inicio del mes (L-D)
m => Almacena el mes valido.
agno => Almacena el año del mes valido.
numero_dias = Almacena el número de días que posee el mes.



Incluye los siguientes procedimientos y funciones de carácter privado:

Void ObtenerDatosLocales() => Lee la fecha local y la almacena.

Void CalcularMesesSucesivos() => Calcula los meses validos siguientes y los almacena.

Int CalcularInicioMes(int mes, int agno) => Calcula el día de semana por el que empieza cada mes.
Bool EsBisiesto(int agno) => Comprueba si ese año es bisiesto.
Int CalcularNuemeroDias(int mes, int agno) => Calcula el número de días del mes.

CalendarioMes

Posee un vector de matrices de tamaño [31][horas_dia+2] de valores enumerados.
31 → máximo de días/mes, horas_dia (12 según especificaciones pedidas, +2 campos como "centinelas" del estado del dia (lleno total, parcial).
El tipo enumerado consta de seis posibles valores.
Libre, ocupado, lleno, fin_semana, pasado (anterior a la fecha actual), no valido (días 31,etc de meses cortos).

Ejem febrero 2017.

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	0	0	0	5

Fin de semana

Fin de semana

Completo

Fin de semana

Parcial

Fin de semana

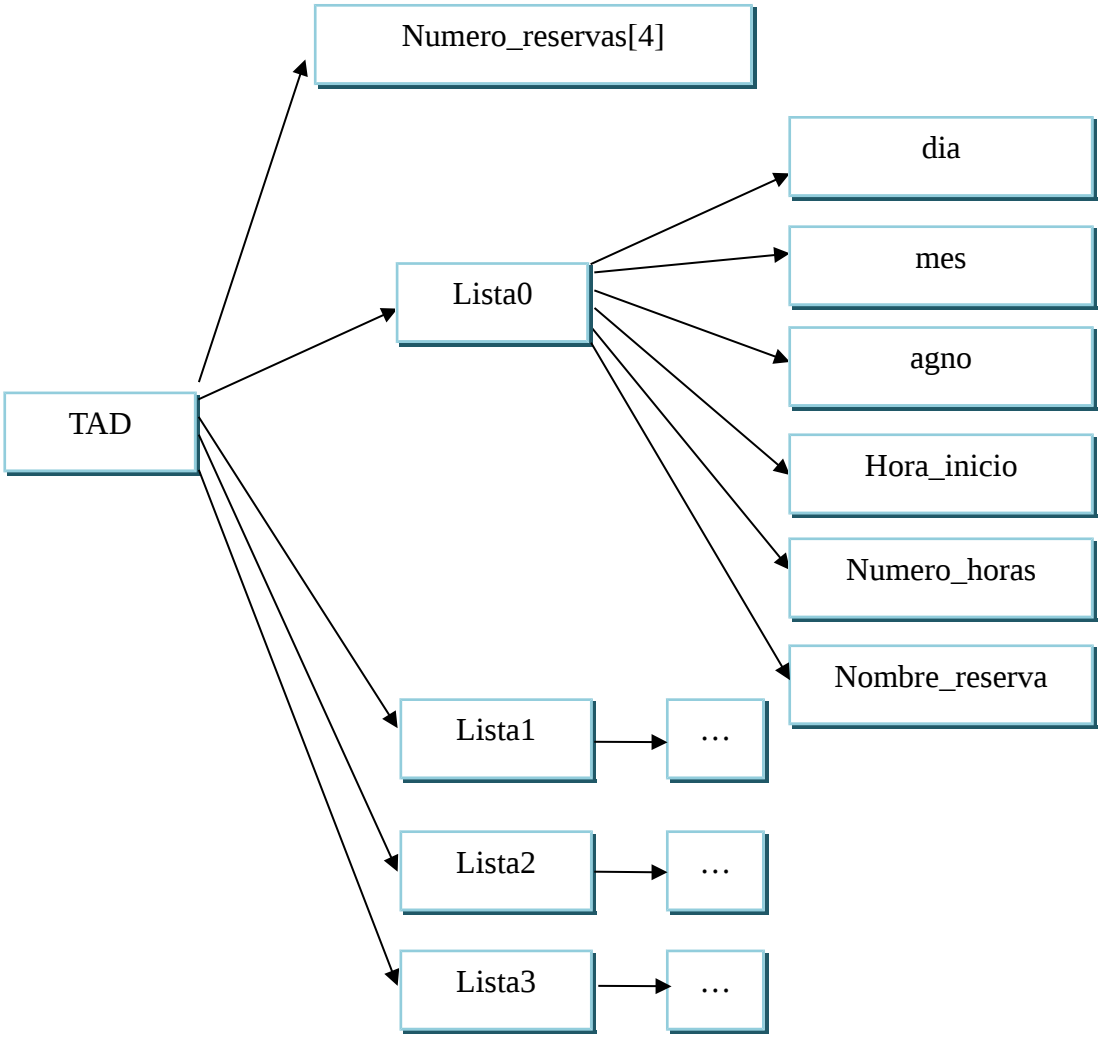
Días no validos

Dispone de dos argumentos:
Void inicializar(argumentos) => Inicializa las matrices y anota fines de semana, etc.
Void cambios_reserva(argumetos) => Actualiza la matriz por cada reserva nueva/anulada

GestionReservasSala.

Posee un vector de tamaño [numero_meses] donde se almacenan el total de reservas/mes.
Dispone de cuatro listas dinámicas (una por mes) donde se almacenan seis datos.

Dia => dia de la reserva
Mes => mes de la reserva
Agno => año de la reserva
Hora_inicio => hora de comienzo de la reserva.
Numero_horas => horas de duración.
Nombre_reserva => Nombre de la persona que reserva.



Incluye los siguientes procedimientos y funciones:

Void inicializar_contador_reservas() => Inicializa el vector numero_reservas
Void introduce_nodo() => Introduce una nueva reserva.
Void imprimir_lista_dia() => imprime las reservas para un día dado.
Int borrar_elemento() => Elimina una reserva de una la lista correspondiente.

Diseño de la practica

Algoritmo general.

Para le realización del programa se han usado los diferentes TAD ya indicados mas un modulo llamado **configuracion** donde se han introducido las constantes de configuración (numero de meses, numero de horas al dia, etc) para facilitar la modificación del programa.
Desgraciadamente el manual de estilo prohíbe las formaciones con punteros por lo cual no se han realizado arreglos de listas. Esto implica que para modificar el numero de meses no solo hay que cambiar la constante numero_meses si no unas cuantas lineas de código.
Ejem →

GestionRerervaSala.h habría que añadir tantas listas como meses adicionales en la linea 36

practica_4.cpp habría que añadir casos (uno por mes adiciona) al switch en las lineas 171-189
etc.

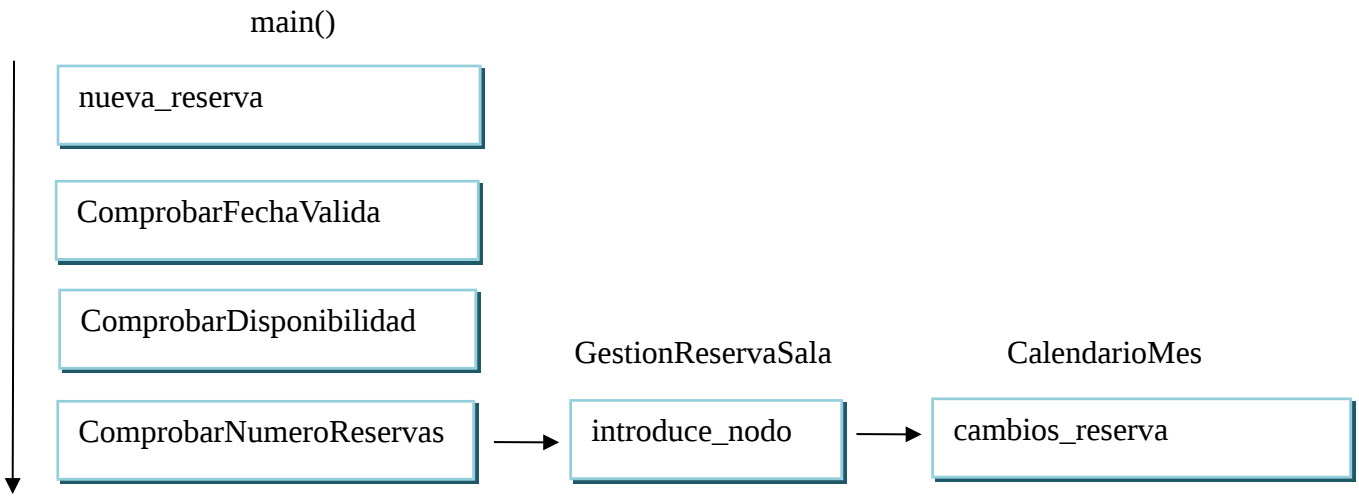
Este problema se podría haber solucionado haciendo un arreglo de tamaño [numero_meses] [maximoreservasmes] de estructuras.
Esta solución implica mayor numero de procesos en la ordenación de elementos, borrado,etc, mayor cantidad de lineas de código y por tanto menor legibilidad, y la reserva de memoria de todas las posibles reservas independientemente de si hay 1 u 80.

La otra posible solución hubiese sido usar una única lista donde almacenar todas las reservas independientemente del numero de meses. Esta podría originar mayor dificultad en su manejo y por tanto errores inesperados.

Por lo expuesto arriba se ha sacrificado la facilidad de ampliación en el registro de meses a cambio de mayor funcionalidad, facilidad en el código y evitar errores no contemplados.

En el archivo principal se han incluido tanto el menú principal como los secundarios, encargándose de la obtención de datos y la verificación de los mismos llamando a diferentes procesos y funciones para ello. Si durante la verificación se obtiene un error el proceso se detiene y muestra por pantalla el error dado. Si las verificaciones son correctas llama a los procedimientos propios de cada TAD y estos realizan los cambios.

Ejem.

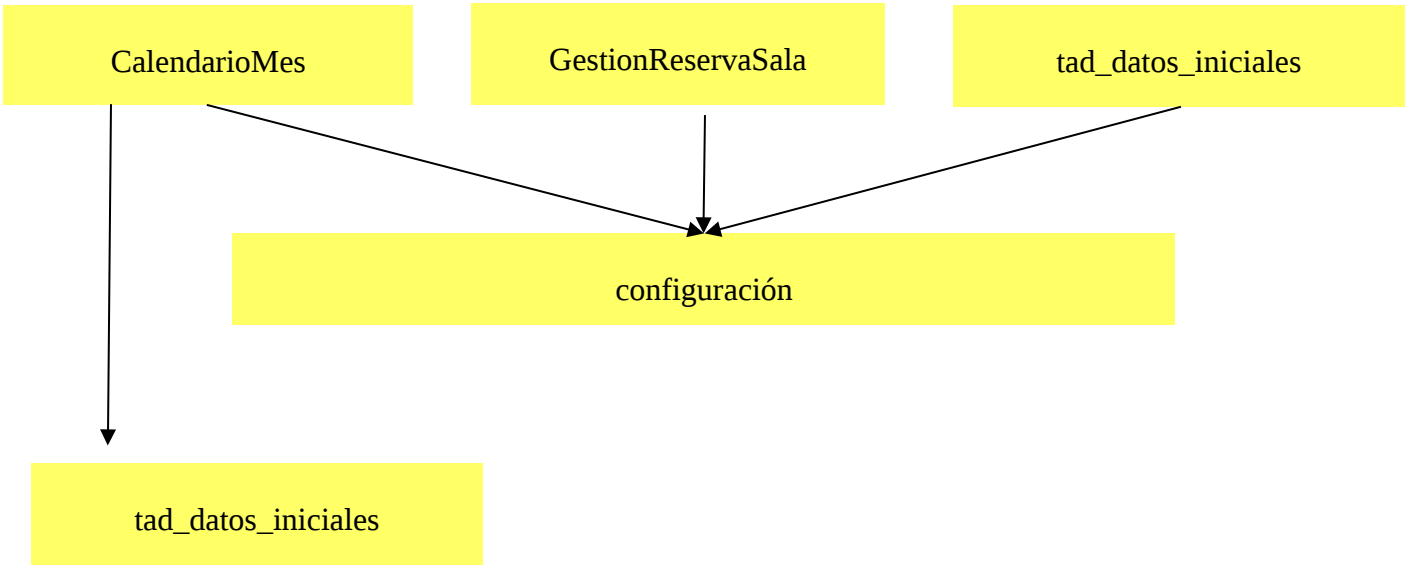

main()
nueva_reserva
ComprobarFechaValida
ComprobarDisponibilidad
ComprobarNumeroReservas
GestionReservaSala
introduce_nodo
CalendarioMes
cambios_reserva

Funciones y procedimientos definidos en archivo principal.

Int ComprobarDisponibilidad (argumentos)	→	Comprueba que las horas sean correctas y si lo son, que estén libres. Si esta fuera de rango devuelve -1, si esta ocupada o no disponible devuelve 0, si esta libre devuelve -10.
int ComprobarNumeroReservas (argumentos)	→	Comprueba que las reservas del mes sean <20 Devuelve 1 si esta disponible 0 si se ha llegado al máximo
int ComprobarFechaValida (argumentos)	→	Comprueba que la fecha sea una fecha valida. Si es valida devuelve el valor de i, que coincide con la posicionen el arreglo. Si no es valida devuelve -1.
void nueva_reserva (argumentos)	→	Recoge los datos de la nueva reserva y va llamando a las funciones correspondientes si no existe un error de disponibilidad. En caso de error muestra el error dado y en caso contrario llama a los procedimientos de los TAD implicados.
void anular_reserva (argumentos)	→	Recoge los datos de la reserva a anular y los comprueba. Elimina la reserva si existe o avisa del error. Llama a los procedimientos de los TAD implicados.
void imprimir_calendario(argumentos)	→	Recoge los datos necesarios, comprueba que las fechas sean correctas e imprime el calendario tal y como especifica el enunciado.
void imprimir_dia (argumentos)	→	Procedimiento que lee los datos del día a mostrar comprueba fechas e imprime día y posibles reservas

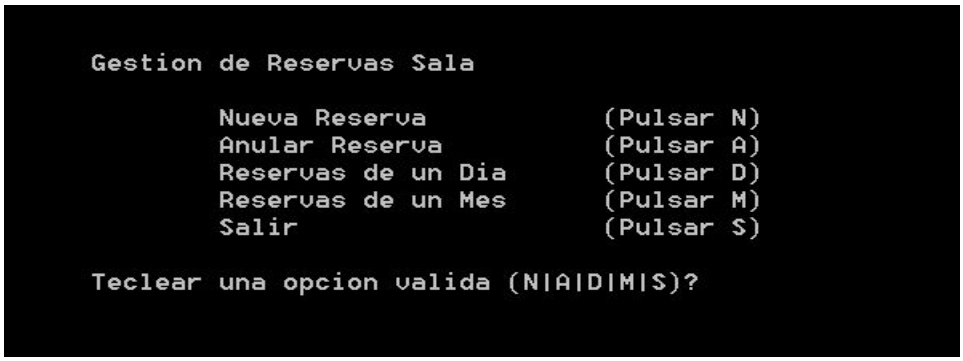
MODULOS:

Prácticamente cada TAD o Modulo son independientes. Todos cargan el modulo configuración y solo CalendarioMes precisa la carga de tad_datos_iniciales



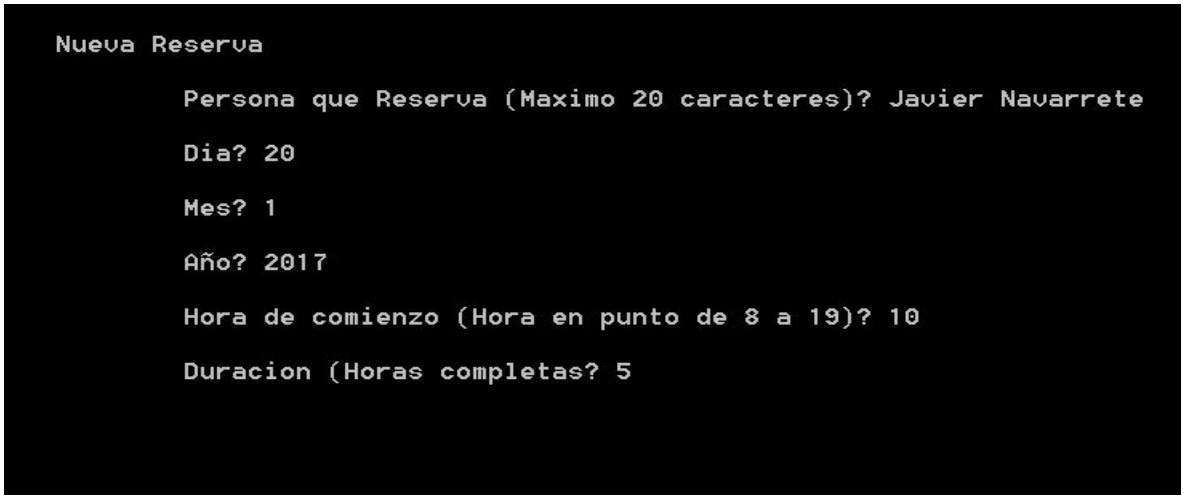
Manual Usuario

Al iniciar el programa aparece el menú principal.



Desde este menú se puede acceder a cada uno de los apartados pulsando la tecla indicada (no distingue entre mayúsculas o minúsculas).
En caso de no pulsar una de las teclas predefinidas la pantalla se refresca y permanece a la espera.

Nueva Reserva.

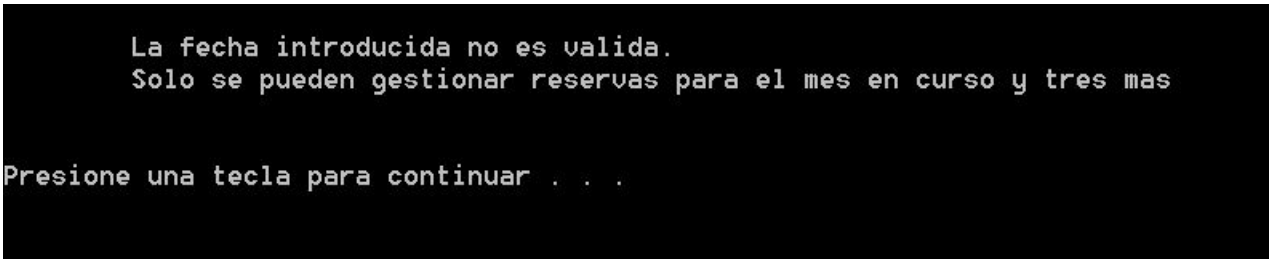


Se irán pidiendo los datos secuencialmente.

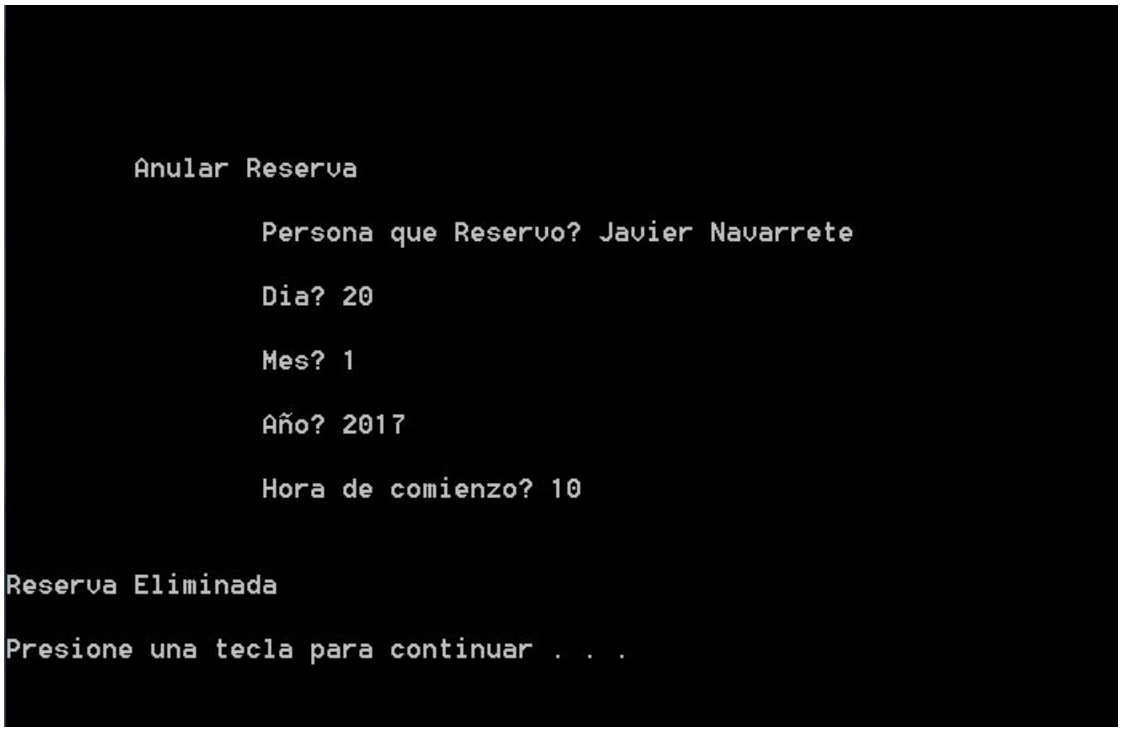
Persona que reserva	→	20 caracteres sensibles a mayúsculas o minúsculas. Acepta espacios en blanco
Día	→	dato de tipo int de 1 a 31.
Mes	→	dato de tipo int de 1 a 12.
Año	→	formato YYYY de tipo int.
Hora de comienzo	→	dato tipo int de 8 a 19.
Duración	→	dato de tipo int. Máximo 12 horas.

En caso de meter una fecha no valida o ya reservada se emitirá el correspondiente mensaje de error. Al pulsar una tecla se regresa al menú principal

ejemplo de error



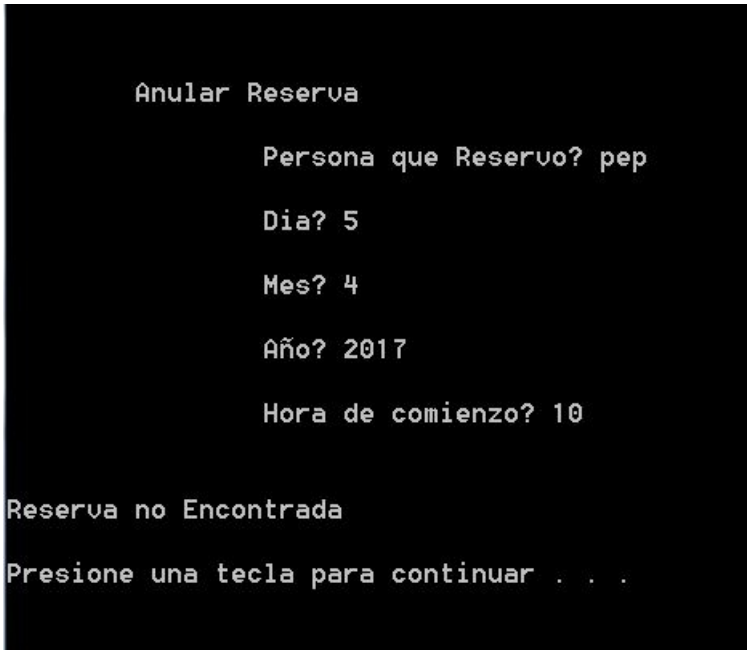
Anular Reserva



Se irán pidiendo los datos secuencialmente.

Persona que reservo	→	20 caracteres sensibles a mayúsculas o minúsculas. Acepta espacios en blanco
Día	→	dato de tipo int de 1 a 31.
Mes	→	dato de tipo int de 1 a 12.
Año	→	formato YYYY de tipo int.
Hora de comienzo	→	dato tipo int de 8 a 19.

En caso de no encontrar la reserva indicada dará el siguiente error.

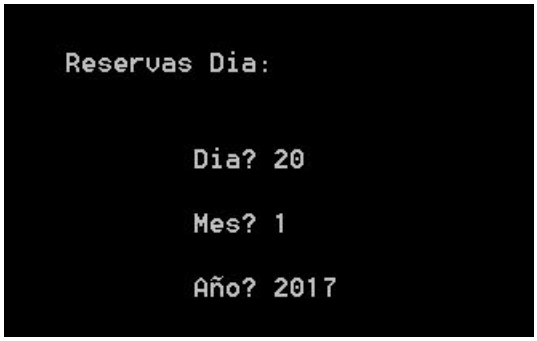


Tras pulsar una tecla se regresa al menú principal.

Si el proceso ha sido satisfactorio se notificara al usuario y tras pulsar una tecla se regresa al menú principal.

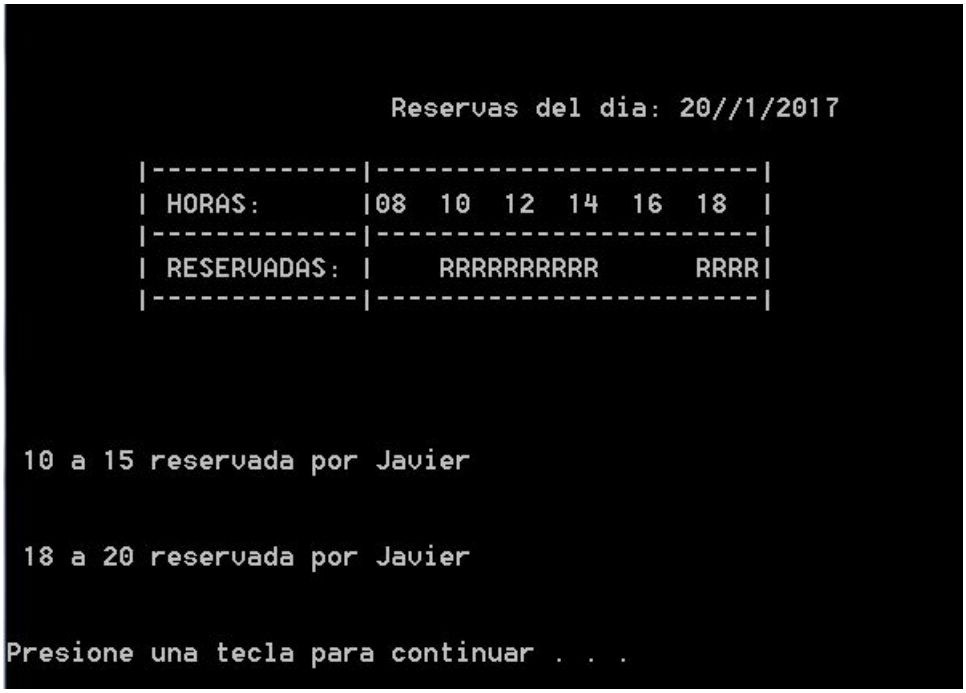
Reservas de un Día.

Se solicitaran los datos necesarios de forma secuencial.



Dia	→	dato de tipo int de 1 a 31.
Mes	→	dato de tipo int de 1 a 12.
Año	→	formato YYYY de tipo int.

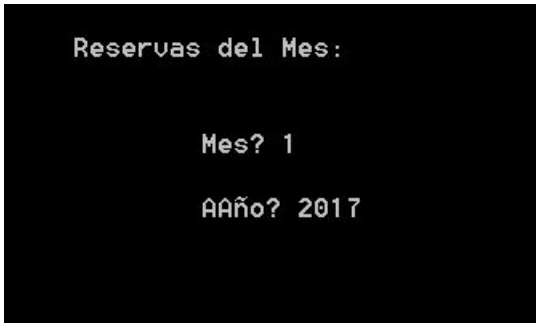
Si la fecha es correcta se mostrara las reservas para el dia indicado.



R= reservado.

Reservas Mes

Se solicitaran los datos necesarios de forma secuencial.



Mes → dato de tipo int de 1 a 12.
Año → formato YYYY de tipo int.

Si los datos son correctos se mostrara el calendario

Si el día esta libre se indicara mostrando el numero de día.
Si hay reservas pero aun quedan horas libres se mostrara PA (parcial)
Si la sala esta reservada todo el día se mostrara TO (total).

Salir

Desde el menú principal pulsar S ó s.

Código Fuente.

Configuración.h

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/

#pragma once

/*****************************************************************************
* en esta interfaz se definen datos generales para facilitar la configuración y
* modificación del programa
*****************************************************************************/
const int numero_meses=4;
const int t_nombre=20;
const int horas_dia=12;
typedef char t_persona_reserva[t_nombre+1];

void pausa();
void borrar_pantalla();
```

configuracion.cpp

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/
#include "configuracion.h"
#include <stdlib.h>
/*****************************************************************
* se implementan dos procedimientos de uso general en el programa
*****************************************************************/

void pausa(){

  system("pause");
}
void borrar_pantalla(){

  system("clear");
  }
```

tad_datos_iniciales.h

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/

#pragma once
#include "configuracion.h"

/*****************************************************************************
* interfaz del tad que almacena los datos iniciales para que sean mas accesibles durante
* le ejecucion del programa. Almacena fecha actual, y en un array de tamaño [numero_mese]
* de estructuras, fecha del mes, numero dias, dia de la seman por el que comienza, etc
*****************************************************************************/
typedef struct t_datos_mes{

  int inicio_semana;
  int m;
  int agno;
  int numero_dias;

};
typedef t_datos_mes meses[numero_meses];

typedef struct t_datos_iniciales{

  void Iniciar();


  int dia_actual;
  int mes_actual;
  int agno_actual;

  meses mes;

  private:

  void ObtenerDatosLocales();
  void CalcularMesesSucesivos();
  int CalcularInicioMes(int mes, int agno);
  bool EsBisiesto(int agno);
  int CalcularNumeroDias(int mes, int agno);


  };
```

tad_datos_iniciales.cpp

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/

#include <time.h>
#include <stdio.h>
#include "tad_datos_iniciales.h"


/*************************************************************
*  procedimiento que obtiene la fecha local
*************************************************************/
void t_datos_iniciales::ObtenerDatosLocales() {

  typedef tm* Mitipotime;



  time_t tiempo;
  Mitipotime s_tiempo;

  tiempo=time(0);
  s_tiempo = localtime(&tiempo);

  agno_actual=s_tiempo->tm_year + 1900;
  mes_actual=s_tiempo->tm_mon + 1;
  dia_actual=s_tiempo->tm_mday;


}

/*****************************************************************************
* funcion que calcula el dia de la seman en la que empeiza el mes
*****************************************************************************/

int t_datos_iniciales::CalcularInicioMes(int mes, int agno) {
  int dia=1;
  int dia_semana;

  int a=(14-mes)/12;        /*Congruencia de Zeller*/
  int y=agno-a;
  int m=mes+(12*a)-2;
  dia_semana=(dia+y+(y/4)-(y/100)+(y/400)+(31*m)/12)%7;
  if (dia_semana==0) { /*Cuando el resto de la division es 0 es Domingo.*/
```

```
   dia_semana=7;
  }


  /*ver return dia_semana-1;*/
  return dia_semana;



}

/*****************************************************************************
* funcion para ver si es un año bisiesto
*****************************************************************************
/

bool t_datos_iniciales::EsBisiesto(int agno) {

  if (agno%4==0 && !(agno%100==0 && agno%400!=0)) {
    return true;
  }
  return false;
}

/
*****************************************************************************
*
* funcion que evuelve el numero de dias que tiene el mes
*****************************************************************************
**/
int t_datos_iniciales::CalcularNumeroDias(int mes, int agno) {


  switch (mes) {

  case 4:
  case 6:
  case 10:
  case 11:
    return 30;
    break;

  case 2:
    if (EsBisiesto(agno)) {
      return 29;
    } else {
      return 28;
    }
    break;
```

```
  default:
    return 31;
  }


}
/
**********************************************************************************
**
* procedimiento que calcula los meses sucesivos para almecenarlos como fecha valida
**********************************************************************************
***/
void t_datos_iniciales::CalcularMesesSucesivos() {

  for (int i=0; i<numero_meses;i++) {

    if (mes_actual+i>12) {
      mes[i].m= (mes_actual+i)%12;
      mes[i].agno=agno_actual+1;
    } else {
      mes[i].m = mes_actual+i;
      mes[i].agno=agno_actual;
    }
    mes[i].inicio_semana=CalcularInicioMes(mes[i].m, mes[i].agno);
    mes[i].numero_dias=CalcularNumeroDias(mes[i].m, mes[i].agno);



  }

}
/*************************************************************************
* procedimiento que inicializa los datos del TAD
*************************************************************************/
void t_datos_iniciales::Iniciar() {

  ObtenerDatosLocales();
  CalcularMesesSucesivos();

}
```

CalendarioMes.h

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/

#pragma once
#include "tad_datos_iniciales.h"
#include "configuracion.h"

/
*****************************************************************************
****
* interfaz CalendarioMes.h. Se define un vector[numero_meses] de matrices de
* 31 dias por [horas_dia+2]. Siendo los dos ultimos int de cada dia los que indican
* la situacion del dia (si esta libre, hay reservas, es fin de semana, etc, y los
*****************************************************************************
*****/

typedef enum valores {libre,ocupado,lleno,fin_semana,pasado,no_valido};
typedef valores T_matriz[31][horas_dia+2];
typedef T_matriz vector_matrices[numero_meses];

typedef struct datos_calendario{

void inicializar(t_datos_iniciales datos_ini);
void cambios_reserva(int posicion,
                     int dia,
                     int mes,
                     int agno,
                     int hora_ini,
                     int n_horas,
                     valores parametro);
vector_matrices matriz_mes;

private:
void Reajustar_estado_dia(int posicion,int dia);

};
```

CalendarioMes.cpp

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/
#include "CalendarioMes.h"
#include <stdio.h>
#include "configuracion.h"

/*****************************************************************************
procedimiento que reajusta los valores de las posiciones horas_dia y horas_dia+1
de las respectivas matriz mes
*****************************************************************************/
void datos_calendario::Reajustar_estado_dia(int posicion,int dia) {

  int hay_ceros=0;
  int hay_unos=0;

  for (int i =0;i<12;i++) {

  if (matriz_mes[posicion][dia-1][i]==libre) {
      hay_ceros=1;
    } else if (matriz_mes[posicion][dia-1][i]==ocupado) {
      hay_unos=1;
    }
  }
  if(hay_ceros==1 && hay_unos==1){
    matriz_mes[posicion][dia-1][horas_dia]=ocupado;
    matriz_mes[posicion][dia-1][horas_dia+1]=libre;
    }else if(hay_ceros==1 && hay_unos==0){
    matriz_mes[posicion][dia-1][horas_dia]=libre;
    matriz_mes[posicion][dia-1][horas_dia+1]=libre;
    }else if (hay_ceros==0 && hay_unos==1){
    matriz_mes[posicion][dia-1][horas_dia]=ocupado;
    matriz_mes[posicion][dia-1][horas_dia+1]=ocupado;
    }
}
/*****************************************************************************
* procecimento que inserta la nueva reserva en la respectiva matriz mes
*****************************************************************************/
void datos_calendario::cambios_reserva(int posicion,
                                       int dia,
                                       int mes,
                                       int agno,
                                       int hora_ini,
```

```
                        int n_horas,
                        valores parametro) {

  int ajuste_horas, hora_final;
  ajuste_horas=hora_ini-8;/*da posicion en la matriz (8am=posicion0)*/
  hora_final=ajuste_horas + n_horas;

   for (int i=ajuste_horas; i<hora_final;i++) {
     matriz_mes[posicion][dia-1][i]=parametro;
  }

  Reajustar_estado_dia(posicion,dia);

}
/**************************************************************************
procediemiento para inicializar todos los valores de las matrices meses
**************************************************************************/

void datos_calendario::inicializar(t_datos_iniciales datos_ini) {

  int dia_comienzo;

  /*inicializar todos los valores a libre*/

  for (int i=0;i<numero_meses;i++) {
    for (int j=0;j<31;j++) {
      for (int k=0; k<horas_dia+2;k++) {
        matriz_mes[i][j][k]=libre;
      }
    }
  }
  /* fin inicializar todos los valores a libre*/

  for (int i=0;i<numero_meses; i++) {

    if (i==0) {

      dia_comienzo=datos_ini.mes[i].inicio_semana;

      for (int j=0;j<datos_ini.mes[i].numero_dias;j++) {

        if ( j<datos_ini.dia_actual-1) {
          matriz_mes[i][j][horas_dia+1]=pasado;
        }
        if (dia_comienzo==6 || dia_comienzo==7) {

          if (matriz_mes[i][j][horas_dia+1]!=pasado) {
            matriz_mes[i][j][horas_dia+1]=fin_semana;
          }
        }
```

```
        dia_comienzo++;
        if (dia_comienzo==8) {
          dia_comienzo=1;
        }

      }

      /* h para los dias que no existen en el mes*/
      for (int h= datos_ini.mes[i].numero_dias; h<31;h++) {

        matriz_mes[i][h][horas_dia+1]=no_valido;
      }


      /* meses siguientes a actual*/
    } else {
      dia_comienzo=datos_ini.mes[i].inicio_semana;

      for (int j=0;j<datos_ini.mes[i].numero_dias;j++) {


        if (dia_comienzo==6 || dia_comienzo==7) {
          matriz_mes[i][j][horas_dia+1]=fin_semana;
          dia_comienzo++;
          if (dia_comienzo==8) {
            dia_comienzo=1;
          }
        } else {
          dia_comienzo++;
          if (dia_comienzo==8) {
            dia_comienzo=1;

          }
        }

        /* h para los dias que no existen en el mes*/
        for (int h= datos_ini.mes[i].numero_dias; h<31;h++) {

          matriz_mes[i][h][horas_dia+1]=no_valido;
        }



      }

    }
  }
}
```

GestionReservaMes.h

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/
#pragma once
#include "configuracion.h"
/*****************************************************************************
* definicion de tipos y estructuras para obtener como resultado:
* un arreglo de tamaño [numero_meses] donde se almacenan el numero de reservas/mes
* se define la estructuras de las listas donde se almacenan las reservas
* Nota: el manueal de estilo no permite formaciones con punteros,
*      de haberse permitido se hubiese creaado un array de [numero_meses] para que en
*      caso de ampliacion de meses hubiese bastado con cambiar la constante de ambito global
*****************************************************************************/
typedef struct TipoNodo {
  int dia,mes,agno,hora_inicio,numero_horas;
  t_persona_reserva nombre_reserva;
  TipoNodo * siguiente;
};
typedef TipoNodo * TipoSecuencia;
typedef int array_int[numero_meses];

typedef struct reserva_sala {

  array_int numero_reservas;
  TipoSecuencia lista0,lista1,lista2,lista3;

  void incializar_contador_reservas();
  void Introduce_nodo(TipoSecuencia &lista,
                      int dia,
                      int mes,
                      int agno,
                      int hora_inicio,
                      int numero_horas,
                      t_persona_reserva nombre_reserva);

  void imprimir_lista(TipoSecuencia lista);
  int borrar_elemento(TipoSecuencia &lista,
                      int dia,
                      int mes,
                      int agno,
                      int hora_inicio,
                      t_persona_reserva nombre_reserva);
   void imprimir_lista_dia(TipoSecuencia lista, int dia);

  };
```

GestionReservasMes.cpp

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/

#include "GestionReservaSala.h"
#include "configuracion.h"
#include <stdio.h>
#include <string.h>

/*****************************************************************************
* procedimiento que inicializa los valores de array numero_reservas a 0
*(reservas actuales por mes)
*****************************************************************************/
void reserva_sala::incializar_contador_reservas() {

  for (int i=0; i < numero_meses;i++) {
    numero_reservas[i]=0;
  }
}
/* este procedimiento no lo uso luego creo*/
void reserva_sala::imprimir_lista(TipoSecuencia lista) {


  TipoSecuencia cursor;
  cursor = lista;
  //anterior = NULL;
  while (cursor != NULL) {
    printf("\n %d",cursor->dia);
    printf("\n %d",cursor->mes);
    printf("\n %d",cursor->agno);
    printf("\n %d",cursor->hora_inicio);
    printf("\n %d",cursor->numero_horas);
    printf("\n %s",cursor->nombre_reserva);
    printf("\n\n\n");

    cursor=cursor->siguiente;


  }
}
```

```
/*****************************************************************
* procedimiento que inserta una nueva reserva en las listas
*****************************************************************/
void reserva_sala::Introduce_nodo(TipoSecuencia &lista,
                          int dia,
                          int mes,
                          int agno,
                          int hora_inicio,
                          int numero_horas,
                          t_persona_reserva nombre_reserva) {


  TipoSecuencia cursor, nuevo, anterior;
  /*creamos el nuevo nodo*/

  nuevo= new TipoNodo;
  nuevo->dia=dia;
  nuevo->mes=mes;
  nuevo->agno=agno;
  nuevo->hora_inicio=hora_inicio;
  nuevo->numero_horas=numero_horas;
  strcpy(nuevo->nombre_reserva,nombre_reserva);/* strcpy*/

  cursor = lista;
  anterior=NULL;

  while (cursor !=NULL&&cursor->dia<dia) {

    anterior=cursor;
    cursor=cursor->siguiente;
  }


  /*ordenar por hora de inicio*/
  while (cursor !=NULL && cursor->dia == dia && cursor->hora_inicio < hora_inicio) {
    anterior=cursor;
    cursor=cursor->siguiente;
  }

  /*insertamos en secuencia*/
  if (anterior==NULL) {
    nuevo->siguiente=NULL;
    lista= nuevo;
  } else {
    nuevo->siguiente=anterior->siguiente;
    anterior->siguiente=nuevo;
  }

}
```

```
/*****************************************************************************
* funcion que borra el nodo donde se ha almacenado la reserva.
* devuelve el numero de horas (dato no cogido en el formulario) para modificar la
* matriz mes
*****************************************************************************/
int reserva_sala::borrar_elemento(TipoSecuencia &lista,
                          int dia,
                          int mes,
                          int agno,
                          int hora_inicio,
                          t_persona_reserva nombre_reserva) {

 TipoSecuencia cursor,anterior;
 int n_horas=0;


 cursor=lista;
 anterior=NULL;


 while (cursor !=NULL && (cursor->dia!=dia || cursor->mes!=mes || cursor->agno!=agno ||
                    cursor->hora_inicio!=hora_inicio ||
                    strcmp(cursor->nombre_reserva,nombre_reserva)!=0)) {


  anterior=cursor;
  cursor=cursor->siguiente;
 }

 if (cursor !=NULL) {
  n_horas= cursor->numero_horas;

  if (cursor->siguiente==NULL) {



    if (anterior !=NULL) {
     anterior->siguiente=NULL;
    } else {
     lista=NULL;
    }

    delete cursor;

  } else {

    if (anterior==NULL) {

      lista=cursor->siguiente;
      delete cursor;
```

```
      } else {

        anterior->siguiente=cursor->siguiente;
        delete cursor;

      }
    }

    return n_horas;

  } else {


    return 0;
  }

}
/*****************************************************************************
* procedimiento que busca las reservas del dia y si existen las imprime
*****************************************************************************/
void reserva_sala::imprimir_lista_dia(TipoSecuencia lista, int dia) {

  TipoSecuencia cursor;

  cursor = lista;
  //anterior = NULL;
  while (cursor != NULL) {
    if (cursor->dia==dia) {
      printf("\n\n %d a ",cursor->hora_inicio);
      printf("%d ",cursor->hora_inicio + cursor->numero_horas);
      printf("reservada por %s",cursor->nombre_reserva);
      printf("\n");
    }
    cursor=cursor->siguiente;
  }

}
```

practica_4.cpp

```
/*********************************************
* NOMBRE: #javier#
* PRIMER APELLIDO: #navarrete#
* SEGUNDO APELLIDO: #rodriguez#
* DNI: #46893078b#
* EMAIL: #javiernavarreterodriguez@hotmail.com#
*********************************************/

#include <stdio.h>
#include <string.h>
#include "configuracion.h"
#include "tad_datos_iniciales.h"
#include "CalendarioMes.h"
#include "GestionReservaSala.h"

/*****************************************************************
* Funcion que comprueba la disponibilidad de la sala y devuelve
* un int indicando si esta disponible o el codigo de error.
*(dia festivo, dia nulo, dia pasado,etc)
*****************************************************************/
int ComprobarDisponibilidad(int posicion,
                            int dia,
                            int mes,
                            int agno,
                            int hora_ini,
                            int n_horas,
                            datos_calendario reg_meses) {
  int ajuste_horas=hora_ini-8;/*da posicion en la matriz (8am=posicion0)*/
  int hora_final=ajuste_horas+n_horas;
  int tramo_ocupado=0;

  if (ajuste_horas<0 || hora_final>12) {
    return -1;
  } else {

    if ((reg_meses.matriz_mes[posicion][dia-1][horas_dia+1]==2)||
        (reg_meses.matriz_mes[posicion][dia-1][horas_dia+1]==3)||
        (reg_meses.matriz_mes[posicion][dia-1][horas_dia+1]==4)||
        (reg_meses.matriz_mes[posicion][dia-1][horas_dia+1]==5)) {

      return 0;

    } else {
      /*comprobamos el horario*/

      for (int i=ajuste_horas; i<hora_final;i++) {

        if (reg_meses.matriz_mes[posicion][dia-1][i]!=0) {
```

```
          tramo_ocupado=1;
          return 0;
        }
        /*todo correcto*/
        return -10;


      }

    }
  }
}
/**************************************************************************
* funcion que comprueba si se ha alcanzado el numero maximo de resevas (20)
* 1= hay espacio, 0= ya estan las 20 reservas
**************************************************************************/
int ComprobarNumeroReservas(int fechaValida,reserva_sala reservas) {

  if (reservas.numero_reservas[fechaValida]<21) {
    return 1;
  } else {
    return 0;
  }


}
/**************************************************************************
* comprueba que la fecha introducida esta dentro de los cuatro meses (numero_meses)
* permitidos. Devuelve -1 si esta fuera de rango o la posicion que ocupa en los arrays.
**************************************************************************/
int ComprobarFechaValida(int dia, int mes, int agno, t_datos_iniciales datos_ini) {

  /* me sobra dia pte modificar*/
  int i=0;
  while (i<numero_meses) {
    if (datos_ini.mes[i].m==mes && datos_ini.mes[i].agno==agno) {
      return i;
    }
    i++;
  }
  return -1;


}
```

```
/*****************************************************************************
* procedimiento que recoge los datos pertinentes para la nueva reserva.
* LLama a diferentes procedimientos y funciones para comprobar la disponibilidad
* y la grabacion de los datos
*****************************************************************************/
void nueva_reserva(t_datos_iniciales datos_ini,
                   reserva_sala &reservas,
                   datos_calendario &reg_meses) {
  int fechaValida;
  int error_disponibilidad=0;
  t_persona_reserva nombre;
  int dia, mes, agno, hora_ini, n_horas;
  valores parametro;



  borrar_pantalla();


  printf("\n\n\n\n\n");

  printf("\tNueva Reserva\n\n");

  printf("\t\tPersona que Reserva (Maximo 20 caracteres)? ");
  fgets(nombre,t_nombre+1,stdin);/*se utiliza fgets para poder coger
                           un espacio en blanco y ahorrar lineas de un bucle*/

  nombre[strlen(nombre)-1]='\0'; /* cambiamos \n de fgets por un \0 */
  fflush(stdin);

  printf("\n\t\tDia? ");
  scanf("%d",&dia);

  printf("\n\t\tMes? ");
  scanf("%d",&mes);

  printf("\n\t\tA%co? ",164);
  scanf("%d",&agno);

  printf("\n\t\tHora de comienzo (Hora en punto de 8 a 19)? ");
  scanf("%d",&hora_ini);

  printf("\n\t\tDuracion (Horas completas? ");
  scanf("%d",&n_horas);

  fechaValida=ComprobarFechaValida(dia,mes,agno,datos_ini);

  /* Comprobaciones de fecha, disponibilidad, horarios,etc */
  if (fechaValida == -1) {
```

```
    borrar_pantalla();
    printf("\n\tLa fecha introducida no es valida.\n\tSolo se pueden gestionar reservas para el mes en
curso y tres mas\n\n\n");
    error_disponibilidad=1;
    pausa();

  } else if (ComprobarNumeroReservas(fechaValida, reservas)==0) {

    borrar_pantalla();
    printf("\n\tSe ha alcanzado el numero maximo de reservas/mes (20 reservas).\n\tNo se pueden
hacer mas reservas ese mes.\n\n\n");
    error_disponibilidad=1;
    pausa();
  } else if (ComprobarDisponibilidad(fechaValida,dia,mes,agno,hora_ini,n_horas,reg_meses)==0) {
    borrar_pantalla();
    printf("\n\tLa sala no esta disponible en la fecha y hora indicadas.\n\tPor favor, consulte la
disponibilidad de la sala\n\tLe recordamos que los fines de semana la instalaciones permaneceran
cerradas.\n\n\n");
    error_disponibilidad=1;
    pausa();
  } else if (ComprobarDisponibilidad(fechaValida,dia,mes,agno,hora_ini,n_horas,reg_meses)==-1) {
    borrar_pantalla();
    printf("\n\tReserva fuera de horario.\n\tLa disponibilidad de la sala es de 8 a 19 horas.\n\n\n");
    error_disponibilidad=1;
    pausa();
  }
  /* fin de comprobaciones. Iniciamos la insercion si no ha dado ningun error */
  if (error_disponibilidad==0) {


    switch (fechaValida) {

    case 0:
      reservas.Introduce_nodo(reservas.lista0, dia, mes, agno, hora_ini, n_horas, nombre);
      break;
    case 1:
      reservas.Introduce_nodo(reservas.lista1, dia, mes, agno, hora_ini, n_horas,nombre);
      break;
    case 2:
      reservas.Introduce_nodo(reservas.lista2, dia, mes, agno, hora_ini, n_horas,nombre);
      break;
    case 3:
      reservas.Introduce_nodo(reservas.lista3, dia, mes, agno, hora_ini, n_horas,nombre);
      break;
      default:

      printf("Error de fecha");

    }
    reservas.numero_reservas[fechaValida]++;
```

```
    /*modificamos las tablas del mes*/
    reg_meses.cambios_reserva(fechaValida,dia,mes,agno,hora_ini,n_horas,ocupado);

    printf("\n\n\n\tReserva Almacenada\n\n");
   pausa();
  }

}
/*****************************************************************************
* procedimiento que toma los datos de la reserva a anular, los comprueba y los borra
* si son correctos
*****************************************************************************/
void anular_reserva(t_datos_iniciales datos_ini,
                    reserva_sala &reservas,
                    datos_calendario &reg_meses) {


  int fechaValida;
  int error_fecha=0;
  t_persona_reserva nombre;
  int dia, mes, agno, hora_ini;
  int n_horas;

borrar_pantalla();


  printf("\n\n\n\n\n");

  printf("\tAnular Reserva\n\n");

  printf("\t\tPersona que Reservo? ");
  fgets(nombre,t_nombre+1,stdin);
  nombre[strlen(nombre)-1]='\0';
  fflush(stdin);

  printf("\n\t\tDia? ");
  scanf("%d",&dia);

  printf("\n\t\tMes? ");
  scanf("%d",&mes);

  printf("\n\t\tA%co? ",164);
  scanf("%d",&agno);

  printf("\n\t\tHora de comienzo? ");
  scanf("%d",&hora_ini);


/* comprobamos fecha y sacamos valor de la posicion y lista*/
```

```
  fechaValida=ComprobarFechaValida(dia,mes,agno,datos_ini);
  if (fechaValida == -1) {

  borrar_pantalla();
    printf("\n\tLa fecha introducida no es valida.\n\tSolo se pueden gestionar reservas para el mes en
curso y tres mas\n\n\n");
    error_fecha=1;
    pausa();

  }

  if (error_fecha==0) {

    switch (fechaValida) {

    case 0:
      n_horas=reservas.borrar_elemento(reservas.lista0, dia, mes, agno, hora_ini,nombre);
      break;
    case 1:
      n_horas=reservas.borrar_elemento(reservas.lista1, dia, mes, agno, hora_ini,nombre);
      break;
    case 2:
      n_horas=reservas.borrar_elemento(reservas.lista2, dia, mes, agno, hora_ini,nombre);
      break;
    case 3:
      n_horas=reservas.borrar_elemento(reservas.lista3, dia, mes, agno, hora_ini, nombre);
      break;
    default:
      printf("Error de fecha");
    }

    if (n_horas!=0){
    reservas.numero_reservas[fechaValida]--;
    reg_meses.cambios_reserva(fechaValida,dia,mes,agno,hora_ini,n_horas,libre);
      printf ("\n\nReserva Eliminada\n\n");
      pausa();
      }else{
      printf("\n\nReserva no Encontrada\n\n");
      pausa();

      }

  }
}
```

```
/*****************************************************************************
* procedimiento que lee los datos del calendario a imprimir, comprueba los datos
* e imprime el calendario
*****************************************************************************/
void imprimir_calendario(t_datos_iniciales datos_ini,
                         reserva_sala reservas,
                         datos_calendario reg_meses) {
  int dia,mes, agno,fechaValida;
  int error_disponibilidad=0;

  /******* variables de practica 3 ********/
  int comienzoSemana;
  int aux;
  typedef char  cadena[26];
  typedef int dias;
  typedef struct T_meses {
    cadena nombre;
  };
  typedef T_meses array [12];
  array meses;

  strcpy(meses[0].nombre,"ENERO                ");
  strcpy(meses[1].nombre,"FEBRERO               ");
  strcpy(meses[2].nombre,"MARZO                ");
  strcpy(meses[3].nombre,"ABRIL               ");
  strcpy(meses[4].nombre,"MAYO                ");
  strcpy(meses[5].nombre,"JUNIO               ");
  strcpy(meses[6].nombre,"JULIO               ");
  strcpy(meses[7].nombre,"AGOSTO                ");
  strcpy(meses[8].nombre,"SEPTIEMBRE              ");
  strcpy(meses[9].nombre,"OCTUBRE               ");
  strcpy(meses[10].nombre,"NOVIEMBRE                ");
  strcpy(meses[11].nombre,"DICIEMBRE              ");



  /******* fin variables pracrica 3 *******/


  borrar_pantalla();

  printf("\n\n\n\n\n");

  printf("\tReservas del Mes: \n\n");

  printf("\n\t\tMes? ");
  scanf("%d",&mes);

  printf("\n\t\tAA%co? ",164);
  scanf("%d",&agno);
```

```
  fechaValida=ComprobarFechaValida(dia,mes,agno,datos_ini);
  if (fechaValida == -1) {

    borrar_pantalla();
    printf("\n\tLa fecha introducida no es valida.\n\tSolo se pueden gestionar reservas para el mes en
curso y tres mas\n\n\n");
    error_disponibilidad=1;
    pausa();
  } else {


    /************************** codigo modificado de la practica 3 ************/


   borrar_pantalla();

    printf("\n\n\n\n");

    comienzoSemana = datos_ini.mes[fechaValida].inicio_semana;


    /*======= CABECERA DEL CALENDARIO ===============*/
    printf("\t\t%s%d\n",meses[mes-1].nombre,agno);
    printf("\t\t%s\n","============================");
    printf("\t\t%s\n","LU  MA  MI  JU  VI | SA  DO");
    printf("\t\t%s\n","============================");
    printf("\t\t");
    if (comienzoSemana != 0) {
      aux = comienzoSemana;

      for ( int i=1;i<comienzoSemana;i++) {

        if (i == 5) {
          printf(" . | ");
        } else {
          printf(" .  ");
        }
        aux++;
        if (aux==7) {
          aux=0;

        }

      }

      for (int k=1; k <= datos_ini.mes[fechaValida].numero_dias;k++) {
```

```
  if (comienzoSemana== 5) {

    if (reg_meses.matriz_mes[fechaValida][k-1][horas_dia+1]==0 &&
        reg_meses.matriz_mes[fechaValida][k-1][horas_dia]==0) {
      printf("%2d | ",k);
    } else if (reg_meses.matriz_mes[fechaValida][k-1][horas_dia+1]==1) {

      printf("TO | ");
    } else if (reg_meses.matriz_mes[fechaValida][k-1][horas_dia]==1) {

      printf("PA | ");
    } else {
      printf("%2d | ",k);
    }
  } else {
    if (reg_meses.matriz_mes[fechaValida][k-1][horas_dia+1]==0 &&
        reg_meses.matriz_mes[fechaValida][k-1][horas_dia]==0) {
      printf("%2d  ",k);
    } else if (reg_meses.matriz_mes[fechaValida][k-1][horas_dia+1]==1) {

      printf("TO  ");
    } else if (reg_meses.matriz_mes[fechaValida][k-1][horas_dia]==1) {

      printf("PA  ");
    } else {
      printf("%2d  ",k);
    }


  }

  comienzoSemana++;
  if (comienzoSemana==8) {
    comienzoSemana=1;
    printf("\n\t\t");
  }

}

while (!(comienzoSemana ==1)) {
  if (comienzoSemana== 5) {
    printf(" . | ");
  } else {
    printf(" .  ");
  }
  comienzoSemana++;
  if (comienzoSemana==8) {
    comienzoSemana=1;

  }
```

```
      }

    }
    printf("\n");

    /************************** fin codigo practica 3 *********/
  }

  printf("\n\n\n\t");
  pausa();

}

/*typedef char t_persona_reserva[t_nombre+1];*//*me la llevo a configuracion*/
/*****************************************************************************
* procedimiento que lee los datos del dia a mostror comprueba fechas e imprime dia
* y posibles reservas
*****************************************************************************/
void imprimir_dia(t_datos_iniciales datos_ini,
                  reserva_sala reservas,
                  datos_calendario reg_meses) {
  int dia, mes, agno;
  int fechaValida;
  int error_disponibilidad=0;
  borrar_pantalla();

  printf("\n\n\n\tReservas Dia:\n\n");

  printf("\n\t\tDia? ");
  scanf("%d",&dia);

  printf("\n\t\tMes? ");
  scanf("%d",&mes);

  printf("\n\t\tA%co? ",164);
  scanf("%d",&agno);

  fechaValida=ComprobarFechaValida(dia,mes,agno,datos_ini);

  /*************************************************************************
  Aqui se realiza las comprobaciones de fecha, disponibilidad, horarios,etc
  *************************************************************************/
  if (fechaValida == -1) {

    borrar_pantalla();
    printf("\n\tLa fecha introducida no es valida.\n\tSolo se pueden gestionar reservas para el mes en
curso y tres mas\n\n\n");
    error_disponibilidad=1;
    pausa();
```

```
}
/*fin de comprobaciones. Iniciamos la insercion si no ha dado ningun error*/

if (error_disponibilidad==0) {

 borrar_pantalla();

  printf("\n\n\n\n");

  printf("\t\t\tReservas del dia: %d//%d/%d\n\n",dia,mes, agno);

  printf("\t|-------------|-----------------------|\n");
  printf("\t| HORAS:      |08  10  12  14  16  18  |\n");
  printf("\t|-------------|-----------------------|\n");
  printf("\t| RESERVADAS: |");
  for (int i=0;i<horas_dia;i++) {

    if (reg_meses.matriz_mes[fechaValida][dia-1][i]==0) {
      printf("   ");
    } else {
      printf("RR");
    }

  }
  printf("|\n");
  printf("\t|-------------|-----------------------|\n\n\n");


  switch (fechaValida) {

  case 0:
    reservas.imprimir_lista_dia(reservas.lista0,dia);
    break;
  case 1:
    reservas.imprimir_lista_dia(reservas.lista1,dia);
    break;
  case 2:
    reservas.imprimir_lista_dia(reservas.lista2,dia);
    break;
  case 3:
    reservas.imprimir_lista_dia(reservas.lista3,dia);
    break;
  default:
  printf("Error de fecha");
  }
  printf ("\n\n");
pausa();

}
```

```
}

int main() {


  t_datos_iniciales datos_ini;
  datos_calendario reg_meses;
  reserva_sala reservas;
  char tecla_ppal=' ';


  reservas.lista0=NULL;
  reservas.lista1=NULL;
  reservas.lista2=NULL;
  reservas.lista3=NULL;


  reservas.incializar_contador_reservas();
  datos_ini.Iniciar();
  reg_meses.inicializar(datos_ini);

  /* menu principal*/

  while (tecla_ppal ==' ') {

  borrar_pantalla();
    fflush(stdin);


    printf("\n\n\n\n\n");

    printf("\tGestion de Reservas Sala\n\n");

    printf("\t\tNueva Reserva\t\t(Pulsar N)\n");

    printf("\t\tAnular Reserva\t\t(Pulsar A)\n");

    printf("\t\tReservas de un Dia\t(Pulsar D)\n");

    printf("\t\tReservas de un Mes\t(Pulsar M)\n");

    printf("\t\tSalir\t\t\t(Pulsar S)\n\n");

    printf("\tTeclear una opcion valida (N|A|D|M|S)?  ");
    scanf("%c",&tecla_ppal);
    fflush(stdin);

    if (tecla_ppal=='n' || tecla_ppal=='N') {
      nueva_reserva(datos_ini,reservas,reg_meses);
      tecla_ppal=' ';
```

```
    } else if (tecla_ppal=='a' || tecla_ppal=='A') {

      anular_reserva(datos_ini,reservas,reg_meses);
      tecla_ppal=' ';
    } else if (tecla_ppal=='d' || tecla_ppal=='D') {
      imprimir_dia(datos_ini,reservas,reg_meses);

      tecla_ppal=' ';
    } else if (tecla_ppal=='m' || tecla_ppal=='M') {
      imprimir_calendario(datos_ini,reservas,reg_meses);
      tecla_ppal=' ';
    } else if (tecla_ppal=='s' || tecla_ppal=='S') {
      /*salimos*/
    } else {
      tecla_ppal=' ';
    }

  }

}
```