Universidad Nacional de Educación a Distancia

Grado en Ingeniería Informática

Práctica 4

Curso 2024-2025

Fundamentos de Programación

Natalia Solana Gavrilenko

DNI:54866052N

Índice

1 Objetivo

El objetivo de la práctica es desarrollar un programa en C+- para gestionar las reservas de apartamentos en un edificio.

- Editar Edificio
- Listar Edificios
- Apartamentos Disponibles
- Reservar Apartamento
- Reservas Mensuales de Apartamento

El programa se divide en tres módulos, GesRAE, calendario y menu, cada uno de ellos compuesto por una interfaz ".h " y un fichero de implementación ".cpp".

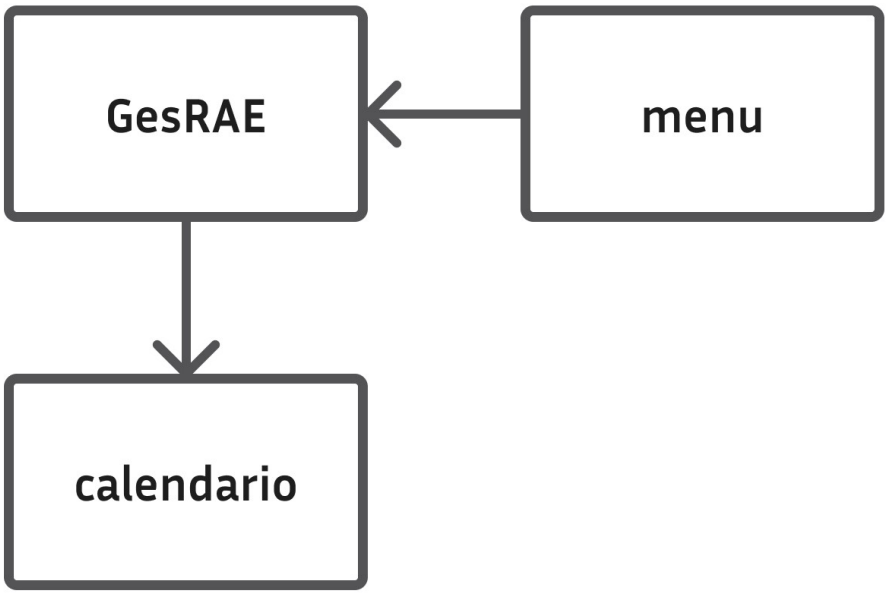

Figure 1: La figura muestra la estructura del programa, indicando las relaciones de dependencia de unos módulos respecto a otros. Las flechas indican que un módulo utiliza directamente elementos de otro.

2 Módulo GesRAE

Este módulo contiene los métodos principales del programa:

2.1 Interfaz

Define las constantes detalladas en el enunciado de la práctica. El máximo numero de Edificios y de Apartamentos.

```
1     const int maxEdifs = 5;
2     const int maxAparts = 20;
```

Seguidamente, se crea un vector de caracteres para poder definir elementos de tipo String. Se eligen 9 caracteres como máximo.

```
1     typedef char tipoString[9];
```

Se define un enumerado con los tipo de Apartamentos que puede tener un edificio:

```
1     typedef enum TipoTipoApart {
2       Basico, Normal, Lujo
3     };
```

Se definen las tres estructuras principales del programa, **TipoReserva, TipoApart y TipoEdif**. La estructura **TipoReserva** incluye un **TipoFecha** (definido en la interfaz calendario) que designa la fecha de entrada a la reserva, un int con la duración de la estancia y un booleano que establece si un apartamento ha sido reservado o no. La estructura **TipoApart** contiene los datos de un apartamento. El identificador de su edificio, un String con la referencia del apartamento, el tipo de apartamento y una lista de reservas (se elige diez como el máximo numero de reservas que puede tener un apartamento). La estructura **TipoEdif** contiene los datos de un edificio. Su identificador y nombre, el numero de apartamentos de cada tipo, el numero de apartamentos en total y el numero de apartamentos libres de cada tipo. Incluye también una lista con todos los apartamentos del edificio.

```
1     typedef struct TipoReserva{
2       TipoFecha fechaEntrada;
3       int duracion;
4       bool reservado;
5     };
6
7     typedef struct TipoApart{
8       int idEdif;
9       tipoString refApart;
10      TipoTipoApart tipoApart;
11
12      TipoReserva listaReservas[10];
13    };
14
```

```
15    typedef struct TipoEdif{
16      int idEdif;
17      int numBasicos;
18      int numNormales;
19      int numLujos;
20      tipoString nombre;
21      int numAparts;
22      int numBasLibres;
23      int numNorLibres;
24      int numLujLibres;
25
26      TipoApart listaAparts[maxAparts];
27    };
```

La siguiente estructura contiene la interfaz de los métodos utilizados en GesRAE.cpp.

```
1     typedef struct GesRAE{
2       void init();
3       int buscarEdif(int ident);
4       void editarEdif(int idEdif, tipoString nombre,
5       int basicos, int normales, int lujos);
6       void listarEdifs();
7       void listarApartsLibres(int idEdif, int dia1, int mes1, int anno1, int duracion1);
8       void reservarApart(int idEdif, TipoTipoApart tipo, int dia1,
9       int mes1, int anno1, int duracion1);
10      void referenciaApart(int idEdif);
11      void reservasMensApart(tipoString Ref, int mes, int anno);
12      bool fechaReal(int dia, int mes, int anno);
13      int diasFecha(int dia, int mes, int anno);
14      bool coincidenciaFechas(int dia1, int mes1, int anno1, int duracion1,
15      int dia2, int mes2, int anno2, int duracion2);
16
17      private:
18      TipoEdif listaEdifs[maxEdifs];
19      int numEdifs;
20      TipoEdif edificio;
21      bool reservado; /*si un apartamento esta reservado en una fecha concreta*/
22      bool completo;
23      /*si todos los apartamentos estan reservados en una fecha concreta*/
24      int numReservas;
25    };
```

2.2 Implementación

A continuación se explica la implementación de cada método.

2.2.1 init

Este método inicializa los datos de la práctica.

```
1     void GesRAE::init(){
2       numReservas=1;
3
4       for(int i = 0; i < maxEdifs; i++ ){
5         listaEdifs[i].numAparts = 0;
6         listaEdifs[i].numBasicos = 0;
7         listaEdifs[i].numNormales = 0;
8         listaEdifs[i].numLujos = 0;
9         listaEdifs[i].numBasLibres = 0;
10        listaEdifs[i].numNorLibres = 0;
11        listaEdifs[i].numLujLibres = 0;
12        listaEdifs[i].idEdif = i + 1;
13        strcpy(listaEdifs[i].nombre, "EdifDeBaja");
14
15        for(int k = 0; k < maxAparts; k ++){
16          strcpy(listaEdifs[i].listaAparts[k].refApart, "APT00X00");
17
18          for(int n = 0; n < 10; n++){
19            listaEdifs[i].listaAparts[k].listaReservas[n].fechaEntrada.dia = 0;
20            listaEdifs[i].listaAparts[k].listaReservas[n].fechaEntrada.mes = 1;
21            listaEdifs[i].listaAparts[k].listaReservas[n].fechaEntrada.anno = 0;
22            listaEdifs[i].listaAparts[k].listaReservas[n].duracion = 0;
23          }
24        }
25      }
26    }
```

2.2.2 buscarEdif

Este método retorna la posición de un edificio según su identificador. Si no se encuentra el edificio, el método retorna -1.

```
1     int GesRAE::buscarEdif( int ident ) {
2       for (int pos = 0; pos < numEdifs; pos++) {
3         if (listaEdifs[pos].idEdif == ident){
4           return pos;
5         }
6       }
7       return -1;
8     }
```

2.2.3 referenciaApart

Crea la referencia de cada apartamento de un edificio. Las referencias de los apartamentos se forman con APT seguido del identificador del edificio, el tipo de apartamento (B-Basico/N-Normal/L-Lujo) y el identificador del apartamento.

```
1   void GesRAE::referenciaApart(int idEdif){
2
3     int pos = buscarEdif(idEdif);
4
5     if(pos != -1){
6       for(int i = 0; i < listaEdifs[pos].numAparts; i++){
7
8         if (listaEdifs[pos].listaAparts[i].tipoApart == Basico) {
9
10          sprintf(listaEdifs[pos].listaAparts[i].refApart,
11           "APT%02dB%02d", idEdif, i + 1);
12        }else{
13          if (listaEdifs[pos].listaAparts[i].tipoApart == Normal) {
14
15            sprintf(listaEdifs[pos].listaAparts[i].refApart,
16             "APT%02dN%02d", pos + 1, i + 1);
17          }else{
18            if (listaEdifs[pos].listaAparts[i].tipoApart == Lujo) {
19
20              sprintf(listaEdifs[pos].listaAparts[i].refApart,
21               "APT%02dL%02d", pos + 1, i + 1);
22            }
23          }
24        }
25      }
26    }
27  }
```

2.2.4 editarEdif

El método editarEdif permite introducir los datos de uno de los 5 edificios.

```
1   void GesRAE::editarEdif(int id, tipoString nom, int basicos, int normales, int lujos){
2     int pos = buscarEdif(id);
3     if(pos != -1){
4       strcpy(listaEdifs[pos].nombre , nom);
5       listaEdifs[pos].numBasicos = basicos;
6       listaEdifs[pos].numNormales = normales;
7       listaEdifs[pos].numLujos = lujos;
8       listaEdifs[pos].numAparts = basicos + normales + lujos;
9
10
```

```
11        for(int i = 0; i<basicos; i++){
12          listaEdifs[pos].listaAparts[i].tipoApart = Basico;
13        }
14
15        for(int i = basicos; i<basicos+normales; i++){
16          listaEdifs[pos].listaAparts[i].tipoApart = Normal;
17        }
18
19        for(int i = basicos+normales; i<basicos+normales+lujos; i++){
20          listaEdifs[pos].listaAparts[i].tipoApart = Lujo;
21        }
22
23        referenciaApart(id);
24      }
25      else{
26        printf("Edificio no encontrado \n");
27      }
28    }
```

2.2.5 listarEdifs

Lista todos los edificios de la lista de edificios con sus datos.

```
1  void GesRAE::listarEdifs(){
2  for (int pos = 0; pos < maxEdifs; pos++){
3    printf("Id     Nombre     Aptos Basicos     Aptos Normales     Aptos de Lujo \n");
4    printf("%d      %s             %d                %d                %d\n" ,
5    pos + 1, listaEdifs[pos].nombre, listaEdifs[pos].numBasicos,
6    listaEdifs[pos].numNormales, listaEdifs[pos].numLujos);
7    }
8  }
```

2.2.6 Métodos Fecha

Los siguientes métodos son necesarios para comprobar si dos reservas coinciden.

- fechaReal

  Este método se asegura de que la fecha introducida es valida.

```
1       bool GesRAE::fechaReal(int dia, int mes, int anno){
2
3         bool real;
4
5         if (anno >= 1601 && anno <= 3000) {
6
7           if(mes >= 1 && mes<= 12){
8
```

```
9
10            if((dia >= 1 && dia <= 28)&&(mes == 2)){
11              real = true;
12            }
13            else if((dia >= 1 && dia <= 30)&&
14            (mes == 4 || mes == 6 || mes == 9 || mes == 11)){
15              real = true;
16            }
17            else if((dia >= 1 && dia <= 31)&&
18            (mes == 1 || mes == 3 || mes == 5 || mes == 7 ||
19            mes == 8 || mes == 10 || mes == 12)){
20              real = true;
21            }
22            else if(dia ==29 && mes == 2&&
23            ((anno % 4 == 0) && (anno % 100 != 0)) || (anno % 400 == 0)){
24              real = true;
25            }
26            else{
27              real = false;
28            }
29          }
30          else{
31            real = false;
32          }
33        }
34        else{
35          real = false;
36        }
37        return real;
38      }
```

- compararFechas

  Este método compara dos fechas para saber cual de ellas es anterior. Devuelve -1 si la primera fecha es aterior, 1 si la segunda fecha es anterior y 0 si son iguales.

```
1       int GesRAE::compararFechas(int dia1, int mes1, int anno1,
2       int dia2, int mes2, int anno2){
3         if(anno1 < anno2){
4           return -1;
5         }else if(anno1 > anno2){
6           return 1;
7         }else{
8           if(mes1 < mes2){
9             return -1;
10          }else if(mes1 > mes2){
11            return 1;
12          }else{
```

```
13            if(dia1 < dia2){
14              return -1;
15            }else if(dia1 > dia2){
16              return 1;
17            }else {
18              return 0;
19            }
20          }
21        }
22      }
```

- diasFecha

  Convierte la fecha introducida en un número de dias.

```
1       int GesRAE::diasFecha(int dia, int mes, int anno){
2
3         int dias = dia;
4         int diasMes;
5
6
7         /*establece el numero de dias de cada mes*/
8         switch(mes){
9           case 4: diasMes = 30;
10          break;
11          case 6: diasMes = 30;
12          break;
13          case 9: diasMes = 30;
14          break;
15          case 11: diasMes = 30;
16          break;
17          case 2:
18          if((anno % 4 == 0 && anno % 100 != 0) || (anno % 400 == 0)){
19            diasMes = 29;
20          }
21          else{
22            diasMes = 28;
23          }
24          break;
25          default: diasMes = 31;
26        }
27        /*suma el dia al numero de dias del mes y lo guarda en dias*/
28        for(int n =1; n<mes; n++){
29          dias = dias + diasMes;
30        }
31
32        /*suma dias al numero de dias en el anno*/
33        for(int a = 1601; a<anno; a++){
```

```
34          if((anno % 4 == 0 && anno % 100 != 0) || (anno % 400 == 0)){
35            dias = dias + 366;
36          }
37          else{
38            dias = dias + 365;
39          }
40        }
41        return dias;
42      }
```

- coincidenciaFechas

  Este método retorna true si dos periodos de reserva se solapan. El primer paso es llamar a compararFechas para saber cual es anterior, una vez establecido; crea dos enteros, uno para la fecha de inicio de las reservas a comparar y otro para la fecha de fin. Si el inicio de la primera fecha es menor que el fin de la segunda y el inicio de la segunda es menor que el fin de la primera, las fechas se solapan. Esta condición solo funciona si la primera fecha es menor que la segunda, por eso es necesario compararlas al principio.

```
1       bool GesRAE::coincidenciaFechas(int dia1, int mes1, int anno1,
2       int duracion1, int dia2, int mes2, int anno2, int duracion2){
3
4         int dia11;
5         int mes11;
6         int anno11;
7
8         int dia22;
9         int mes22;
10        int anno22;
11
12        if(compararFechas(dia1, mes1, anno1, dia2, mes2, anno2)== 1){
13          dia11 = dia2;
14          mes11 = mes2;
15          anno11 = anno2;
16
17          dia22 = dia1;
18          mes22 = mes1;
19          anno22 = anno1;
20        }else if(compararFechas(dia1, mes1, anno1, dia2, mes2, anno2)== -1){
21          dia11 = dia1;
22          mes11 = mes1;
23          anno11 = anno1;
24
25          dia22 = dia2;
26          mes22 = mes2;
27          anno22 = anno2;
28        }else{
29          return true; /*si ambas fechas son la misma los periodos coinciden*/
```

```
30        }
31
32        int inic1 = diasFecha(dia11, mes11, anno11);
33        int fin1 = inic1 + duracion1;
34        int inic2 = diasFecha(dia22, mes22, anno22);
35        int fin2 = inic2 + duracion2;
36
37        return(inic1 < fin2 && inic2 < fin1);
38      }
```

2.2.7 reservarApart

Permite hacer la reserva de un apartamento en para una fecha y duración concreta. Este método hace uso de los métodos explicados anteriormente para comparar fechas.

```
1   void GesRAE:: reservarApart(int idEdif, TipoTipoApart tipo, int dia1,
2    int mes1, int anno1, int duracion1){
3     int pos = buscarEdif(idEdif);
4
5     int dia2;
6     int mes2;
7     int anno2;
8     int duracion2;
9
10    bool reservado;
11    reservado = false; /*variable para cada apartamento*/
12
13    if(pos != -1){
14
15      edificio = listaEdifs[pos];
16
17      switch(tipo){
18
19        case Basico:
20        completo = true;/*variable para ver si todos los apartamentos estan reservados*/
21
22        for(int i = 0; i < edificio.numAparts; i++){
23          if (edificio.listaAparts[i].tipoApart == Basico){
24
25            /*mira cada una de las reservas de cada apartamento de un edificio
26             y las compara con las dadas por el usuario*/
27            for(int n = 0; n < 10; n++){
28              dia2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia;
29              mes2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes;
30              anno2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno;
31              duracion2 = edificio.listaAparts[i].listaReservas[n].duracion;
32
```

```
33              if(coincidenciaFechas(dia1, mes1, anno1, duracion1,
34               dia2, mes2, anno2, duracion2)== false && reservado == false){
35
36                edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia = dia1;
37                edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes = mes1;
38                edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno = anno1;
39                edificio.listaAparts[i].listaReservas[n].duracion = duracion1;
40
41                printf("Apartamento reservado con exito \n");
42                reservado = true;
43                completo = false; /*hay un apartamento libre*/
44
45                printf("        Datos de la reserva: \n");
46                printf("\n");
47                printf("    Numero de Reserva: %d \n", numReservas);
48                printf("    Edificio: %s (Id=%d)\n", edificio.nombre, idEdif);
49                printf("    Referencia Apartamento: %s \n",
50                 edificio.listaAparts[i].refApart);
51                printf("    Fecha Entrada: %d/%d/%d \n", dia1, mes1, anno1);
52                printf("    Duracion estancia: %d \n", duracion1);
53                numReservas++;
54              }
55            }
56          }
57        }
58        if(completo){
59          printf("Todos los apartamentos Basicos estan reservados
60           en la fecha introducida");
61        }
62        break;
63        case Normal:
64        completo = true;
65
66        for(int i = 0; i < edificio.numAparts; i++){
67
68          if (edificio.listaAparts[i].tipoApart == Normal) {
69
70            for(int n = 0; n < 10; n++){
71              dia2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia;
72              mes2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes;
73              anno2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno;
74              duracion2 = edificio.listaAparts[i].listaReservas[n].duracion;
75
76              if(coincidenciaFechas(dia1, mes1, anno1, duracion1,
77               dia2, mes2, anno2, duracion2)== false && reservado == false){
78                edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia = dia1;
79                edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes = mes1;
```

```
80                edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno = anno1;
81                edificio.listaAparts[i].listaReservas[n].duracion = duracion1;
82
83                printf("Apartamento reservado con exito \n");
84                reservado = true;
85                completo = false;
86
87                printf("        Datos de la reserva: \n");
88                printf("\n");
89                printf("    Numero de Reserva: %d \n", numReservas);
90                printf("    Edificio: %s (Id=%d)\n", edificio.nombre, idEdif);
91                printf("    Referencia Apartamento: %s \n",
92                 edificio.listaAparts[i].refApart);
93                printf("    Fecha Entrada: %d/%d/%d \n", dia1, mes1, anno1);
94                printf("    Duracion estancia: %d \n", duracion1);
95
96                numReservas++;
97              }
98            }
99          }
100       }
101       if(completo){
102         printf("Todos los apartamentos Normales estan reservados
103          en la fecha introducida");
104       }
105       break;
106       case Lujo:
107       completo = true;
108
109       for(int i = 0; i < edificio.numAparts; i++){
110
111         if (edificio.listaAparts[i].tipoApart == Lujo) {
112
113           for(int n = 0; n < 10; n++){
114             dia2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia;
115             mes2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes;
116             anno2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno;
117             duracion2 = edificio.listaAparts[i].listaReservas[n].duracion;
118
119             if(coincidenciaFechas(dia1, mes1, anno1, duracion1,
120              dia2, mes2, anno2, duracion2)== false && reservado == false){
121               edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia = dia1;
122               edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes = mes1;
123               edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno = anno1;
124               edificio.listaAparts[i].listaReservas[n].duracion = duracion1;
125
126               printf("Apartamento reservado con exito \n");
```

```
127                 reservado = true;
128                 completo = false;
129
130                 printf("        Datos de la reserva: \n");
131                 printf("\n");
132                 printf("    Numero de Reserva: %d \n", numReservas);
133                 printf("    Edificio: %s (Id=%d)\n", edificio.nombre, idEdif);
134                 printf("    Referencia Apartamento: %s \n",
135                  edificio.listaAparts[i].refApart);
136                 printf("    Fecha Entrada: %d/%d/%d \n", dia1, mes1, anno1);
137                 printf("    Duracion estancia: %d \n", duracion1);
138
139                 numReservas++;
140               }
141             }
142           }
143         }
144         if(completo){
145           printf("Todos los apartamentos de Lujo estan reservados
146            en la fecha introducida");
147         }
148         break;
149       }
150     }
151     else{
152       printf("Edificio no encontrado \n");
153     }
154   }
```

2.2.8 listarApartsLibres

Por último, este método lista todos los apartamentos disponibles de un edificio.

```
1     oid GesRAE::listarApartsLibres(int idEdif, int dia1, int mes1,
2      int anno1, int duracion1){
3       int pos = buscarEdif(idEdif);
4
5       int dia2;
6       int mes2;
7       int anno2;
8       int duracion2;
9
10      TipoTipoApart tipoApartamento;
11
12      bool libre;
13      libre = false;
14
15      if(pos != -1){/*el edificio existe*/
```

```
16          edificio = listaEdifs[pos];
17
18          edificio.numBasLibres = 0;
19          edificio.numNorLibres = 0;
20          edificio.numLujLibres = 0;
21
22          for(int i = 0; i < edificio.numAparts; i++){
23
24            for(int n = 0; n < 10; n++){
25
26              dia2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.dia;
27              mes2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.mes;
28              anno2 = edificio.listaAparts[i].listaReservas[n].fechaEntrada.anno;
29              duracion2 = edificio.listaAparts[i].listaReservas[n].duracion;
30
31              if(coincidenciaFechas(dia1, mes1, anno1, duracion1,
32               dia2, mes2, anno2, duracion2)== false){
33
34                libre = true;
35              }
36            }
37            if(libre==true){
38
39              tipoApartamento = edificio.listaAparts[i].tipoApart;
40
41              switch(tipoApartamento){
42                case Basico:
43                edificio.numBasLibres++;
44                break;
45                case Normal:
46                edificio.numNorLibres++;
47                break;
48                case Lujo:
49                edificio.numLujLibres++;
50                break;
51              }
52              libre=false;
53            }
54          } /*mira todas las reservas de los apartamentos y cuenta
55           cuantos apartamentos hay libres en esa fecha*/
56
57          printf("El edificio %s desde el %d/%d/%d y %d dias de estancia,
58          tendria disponibles:\n", edificio.nombre, dia1, mes1, anno1, duracion1 );
59          printf("%d apartamentos tipo Basico \n", edificio.numBasLibres);
60          printf("%d apartamentos tipo Normal \n", edificio.numNorLibres);
61          printf("%d apartamentos tipo Lujo \n", edificio.numLujLibres);
62        }
```

```
63        else{
64          printf("Edificio no encontrado \n");
65        }
66      }
67
68
69      void GesRAE::reservasMensApart(tipoString ref, int mes, int anno){
70
71      }
```

3 Módulo Calendario

Este módulo contiene los elementos necesarios para imprimir un calendario (desde el año 1601 hasta el 3000).

3.1 Interfaz

Define un enumerado llamado TipoDia que contiene los días de la semana y una estructura llamada TipoFecha con el día, mes y año.

```
1       typedef enum TipoDia {LU, MA, MI, JUs, VI, SA, DO};
2
3       typedef struct TipoFecha {
4         int dia;
5         int mes;
6         int anno;
7       };
```

Sus métodos son los siguientes:

```
1       typedef struct Calendario{
2         bool bisiesto ( int anno );
3         TipoDia sumarDias (TipoDia dia , int n);
4         TipoDia diaSemana ( TipoFecha fecha );
5         int diasDelMes (TipoFecha fecha);
6
7         void imprimirCalendario(int mes, int anno);
8
9         private:
10        int i;
11        TipoFecha fecha;
12        TipoDia primerDia;
13      };
```

3.2 Implementación

A continuación se explica la implementación de cada método.

3.2.1 bisisesto

Comprueba si un Anno es bisiesto o no.

```
1       bool Calendario::bisiesto ( int anno ) {
2         return ((anno % 4 == 0) && (anno % 100 != 0)) || (anno % 400 == 0);
3       }
```

3.2.2 sumarDias

Suma un numero de días a un día y devuelve el día de la semana correspondiente.

```
1       TipoDia Calendario::sumarDias (TipoDia dia , int n) {
2
3         const  int DiasSem = 7;
4         int aux;
5
6         aux = (int (dia) + n ) % DiasSem;
7         return TipoDia (aux);
8       }
```

3.2.3 diasDelMes

Retorna el numero de días de un mes.

```
1       int Calendario::diasDelMes (TipoFecha fecha) {
2         switch (fecha.mes) {
3           case 2:
4           if (bisiesto (fecha.anno)) {
5             return (29);
6           } else { return (28);
7           };
8           break;
9           case 4:
10          case 6:
11          case 9:
12          return (30);
13          break;
14          default:
15          return (31);
16        }
17      }
```

3.2.4 diaSemana

Devuelve el día de la semana de una fecha.

```
1       TipoDia Calendario::diaSemana ( TipoFecha fecha ) {
2
```

```
3         int mes = fecha.mes;
4         int a = fecha.anno;
5         int IncreDias, IncreAnnos, IncreBisies;
6         const int AnnoRef= 1601;
7         const TipoDia DiaReferencia = DO;
8
9         if (mes == 1) {
10          IncreDias = 0;
11        } else if (mes ==2){
12          IncreDias = 3;
13        } else if (mes == 3 ) {
14          IncreDias = 3;
15        } else if (mes == 4 ) {
16          IncreDias = 6;
17        } else if (mes == 5) {
18          IncreDias =   1;
19        } else if (mes == 6) {
20          IncreDias =  4;
21        } else if (mes == 7) {
22          IncreDias = 6;
23        } else if (mes == 8) {
24          IncreDias = 2;
25        } else if (mes == 9) {
26          IncreDias = 5;
27        } else if (mes == 10) {
28          IncreDias = 0;
29        } else if (mes == 11) {
30          IncreDias = 3;
31        } else {
32          IncreDias = 5;
33        }
34
35        IncreAnnos = a - AnnoRef;
36        IncreBisies = 0;
37
38        /*calcula cuantos annos bisiestos ha habido desde el 1601*/
39        for (int n=1602; n<a; n++) {
40          if (bisiesto(n)) {
41            IncreBisies ++;
42          }
43        }
44
45        /*si el anno es bisiesto y el mes posterior a feb incrementa IncreDias*/
46        if ( bisiesto(a) && ( mes > 2 )) {
47          IncreDias ++;
48        }
49
```

```
50          IncreDias = IncreDias + IncreAnnos + IncreBisies + fecha.dia;
51
52          /*calcula el dia de la semana sumando el incremento*/
53          return sumarDias(DiaReferencia, IncreDias);
54        }
```

3.2.5 imprimirCalendario

Finalmente, este método imprime por pantalla el calendario

```
1       void Calendario::imprimirCalendario(int mes, int anno) {
2
3         if(mes>=1 && mes<=12 && anno>=1601 && anno<=3000){
4
5           fecha.dia = 1;
6           fecha.mes = mes;
7           fecha.anno = anno;
8
9           primerDia = diaSemana (fecha);
10
11          printf ("\n");
12
13          switch (fecha.mes) {
14            case 1:
15            printf ("ENERO     ");
16            break;
17            case 2:
18            printf ("FEBRERO   ");
19            break;
20            case 3:
21            printf ("MARZO     ");
22            break;
23            case 4:
24            printf ("ABRIL     ");
25            break;
26            case 5:
27            printf ("MAYO      ");
28            break;
29            case 6:
30            printf ("JUNIO     ");
31            break;
32            case 7:
33            printf ("JULIO                  %d\n", fecha.anno);
34            break;
35            case 8:
36            printf ("AGOSTO                 %d\n", fecha.anno);
37            break;
38            case 9:
```

```
39              printf ("SEPTIEMBRE             %d\n", fecha.anno);
40              break;
41              case 10:
42              printf ("OCTUBRE                %d\n", fecha.anno);
43              break;
44              case 11:
45              printf ("NOVIEMBRE              %d\n", fecha.anno);
46              break;
47              case 12:
48              printf ("DICIEMBRE              %d\n", fecha.anno);
49              break;
50            }
51
52            printf ("              ");
53            printf ("%d\n", fecha.anno);
54            printf ("============================\n");
55            printf ("LU  MA  MI  JU  VI | SA  DO\n");
56            printf ("============================\n");
57
58            i = 0;
59
60            /*imprime tantos puntos como espacios haya hasta el primer
61             dia de la semana del mes*/
62            for (int k = int (primerDia); k >= 1; k --) {
63              if (i % 7 == 5) {
64                printf ("| ");/*si i es igual a 5, 12, etc,
65                 se imprime una columna, queda impresa en la sexta pos de la semana*/
66              }
67              if ( i %7 != 0 && i % 7 != 5 ) {
68                printf (" ");
69              }
70              printf (" . ");
71              i ++;
72            }
73            /*imprime los dias del mes*/
74            for (int k = 0; k < diasDelMes(fecha); k ++) {
75              if (i != 0) {
76                if (i % 7 == 0) {
77                  printf ("\n");/*al final de la semana se pone un salto de linea*/
78                }
79              }
80              if (i % 7 == 5) {
81                printf ("| ");/*sexta posicion, columna*/
82              }
83              if ( i %7 != 0 && i % 7 != 5 ) {/*no es primera pos de la semana ni sexta*/
84                printf (" ");
85              }
```

```
86              printf ("%2d ", k+1 );/*ocupa dos espacios mas un espacio mas despues*/
87              i ++;
88            }
89
90            while ( i % 7 != 0 ) {
91              if (i % 7 == 5) {
92                printf ("| ");
93              }
94              if ( i %7 != 0 && i % 7 != 5 ) {
95                printf (" ");
96              }
97              printf (" . ");
98              i ++;
99            }
100           printf ("\n");
101         }else{
102           printf("Fecha no valida");
103         }
104       }
105       }
```

4 Módulo menu

4.1 Interfaz

La interfaz del módulo menu contiene los métodos implementados en su fichero .cpp. Este módulo define el menú principal del programa, así como los menús secundarios correspondientes a cada operación.

```
1       typedef struct menu{
2         void inicializar();
3         void volver();
4         void menuPrincipal();
5         void menuEditarEdif();
6         void menuListEdifs();
7         void menuApartsLibres();
8         void menuResApart();
9         void menuResMens();
10
11        private:
12
13        GesRAE gesRAE;
14
15      };
```

4.2 Implementación

A continuación se explica la implementación de cada método.

4.2.1 inicializar

Este método llama al método init de GesRAE, e inicializa los datos del programa.

```
1       void menu::inicializar(){
2         gesRAE.init();
3       }
```

4.2.2 menuPrincipal

Imprime el menú principal en pantalla. Según la tecla que pulse el usuario, el programa llamará a la función correspondiente de GesRAE.

```
1       void menu::menuPrincipal(){
2
3         char input;
4
5         do{
6
7           printf("GesRAE: Gestion de Reservas Apartamentos-Edificios \n");
8           printf("\n");
9           printf("    Editar Edificio                    (Pulsar E) \n");
10          printf("    Listar Edificios                   (Pulsar L) \n");
11          printf("    Apartamentos Disponibles           (Pulsar A) \n");
12          printf("    Reservar Apartamento               (Pulsar R) \n");
13          printf("    Reservas Mensuales Apartamento     (Pulsar M) \n");
14          printf("    Salir                              (Pulsar S) \n");
15          printf("\n");
16          printf("Teclear una opcion valida\n");
17
18          scanf(" %c" , &input);
19
20          switch(input){
21            case 'E':
22            menuEditarEdif();
23            break;
24            case 'L':
25            menuListEdifs();
26            break;
27            case 'A':
28            menuApartsLibres();*/
29            break;
30            case 'R':
31            menuResApart();
```

```
32              break;
33              case 'M':
34              menuResMens();
35              break;
36              case 'S':
37              printf("Saliendo del programa");
38              break;
39              default:
40              printf("opcion no valida");
41            }
42          }while(input != 'S');
43        }
```

4.2.3 menuEditarEdif

Imprime el menú que aparece en pantalla cuando se selecciona la opción Editar Edificio. Tras pedir los datos al usuario, llama a la función correspondiente.

```
1
2     void menu::menuEditarEdif(){
3
4       char control;
5       tipoString nombre;
6       int id, numBasicos, numNormales, numLujos;
7
8       printf("Editar Edificio: \n");
9       printf("    Identificador(numero entre 1 y 5)?");
10      scanf("%d" , &id);
11      printf("\n");
12
13      printf("    Nombre(entre 1 y 20 caracteres)?");
14      scanf("%20s" , &nombre);
15      printf("\n");
16
17      printf("    Numero de apartamentos Basicos?");
18      scanf("%d" , &numBasicos);
19      printf("\n");
20
21      printf("    Numero de apartamentos Normales?");
22      scanf("%d" , &numNormales);
23      printf("\n");
24
25      printf("    Numero de apartamentos de Lujo?");
26      scanf("%d" , &numLujos);
27      printf("\n");
28
29      printf("IMPORTANTE: Esta opcion borra los datos anteriores. \n");
```

```
30        printf("Son correctos sus nuevos datos(S/N)? \n");
31        scanf(" %c" , &control);
32        printf("\n");
33
34        if(control == 'S'){
35          gesRAE.editarEdif(id, nombre, numBasicos, numNormales, numLujos);
36          printf("Los datos se han guardado");
37          menuPrincipal();
38        }else{
39          printf("Los datos no se han guardado\n");
40          menuPrincipal();
41        };
42      }
```

4.2.4 menuListarEdifs

Imprime el menú que aparece en pantalla al seleccionar la opción Listar Edificios y llama a la función correspondiente.

```
1       void menu::menuListEdifs(){
2         gesRAE.listarEdifs();
3         menuPrincipal();
4       }
```

4.2.5 menuApartsLibres

Imprime el menú que aparece en pantalla cuando se selecciona la opción Apartamentos Disponibles. Tras pedir los datos al usuario, llama a la función correspondiente.

```
1
2       void menu::menuApartsLibres(){
3
4         int id, dia, mes, anno, duracion;
5
6         printf("Apartamentos disponibles: \n");
7         printf("    Identificador del edificio?");
8         scanf(" %d", &id);
9         printf("\n");
10
11        printf("    Fecha de Entrada: Dia?");
12        scanf(" %d", &dia);
13        printf("\n");
14
15        printf("    Fecha de Entrada: Mes?");
16        scanf(" %d", &mes);
17        printf("\n");
18
19        printf("    Fecha de Entrada: Anno?");
```

```
20        scanf(" %d", &anno);
21        printf("\n");
22
23        printf("    Dias de duracion de la estancia?");
24        scanf(" %d", &duracion);
25        printf("\n");
26
27        gesRAE.listarApartsLibres(id, dia, mes, anno, duracion);
28
29        menuPrincipal();
30        }
```

4.2.6 menuResApart

Imprime el menú que aparece en pantalla cuando se selecciona la opción para reservar un apartamento. Tras pedir los datos al usuario, llama a la función correspondiente.

```
1         void menu::menuResApart(){
2
3           int id, dia, mes, anno, duracion;
4           char tipo;
5           TipoTipoApart tipoApart;
6
7           printf("Reservar apartamento: \n");
8
9           printf("    Identificador del edificio?");
10          scanf(" %d", &id);
11          printf("\n");
12
13          printf("Tipo de Apartamento (B-Basico/N-Normal/L-Lujo)?");
14          scanf(" %c", &tipo);
15          printf("\n");
16
17          printf("    Fecha de Entrada: Dia?");
18          scanf(" %d", &dia);
19          printf("\n");
20
21          printf("    Fecha de Entrada: Mes?");
22          scanf(" %d", &mes);
23          printf("\n");
24
25          printf("    Fecha de Entrada: Anno?");
26          scanf(" %d", &anno);
27          printf("\n");
28
29          printf("    Dias de duracion de la estancia?");
30          scanf(" %d", &duracion);
```

```
31          printf("\n");
32
33          printf("Es correcta la operacion (S/N)? \n");
34
35          switch(tipo){
36
37            case 'B':
38            tipoApart = Basico;
39            gesRAE.reservarApart(id, tipoApart, dia, mes, anno, duracion);
40            menuPrincipal();
41            break;
42
43            case 'N':
44            tipoApart = Normal;
45            gesRAE.reservarApart(id, tipoApart, dia, mes, anno, duracion);
46            menuPrincipal();
47            break;
48
49            case 'L':
50            tipoApart = Lujo;
51            gesRAE.reservarApart(id, tipoApart, dia, mes, anno, duracion);
52            menuPrincipal();
53            break;
54
55            default:
56            printf("opcion no valida");
57            menuPrincipal();
58
59          }
60        }
```

5 Manual de usuario

Inicialmente, el programa muestra por pantalla el menú principal.

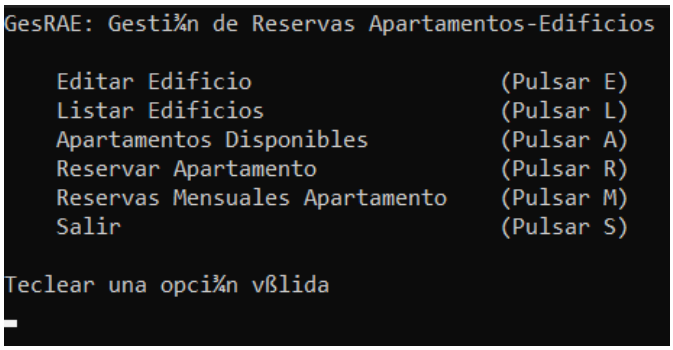

Figure 2: Menú Principal

Al pulsar 'E', se accede al menú de Edición de Edificios. Se pide insertar un Identificador, Nombre y Numero de Apartamentos de cada tipo. Finalmente el usuario tiene que pulsar S para guardar los nuevos datos.

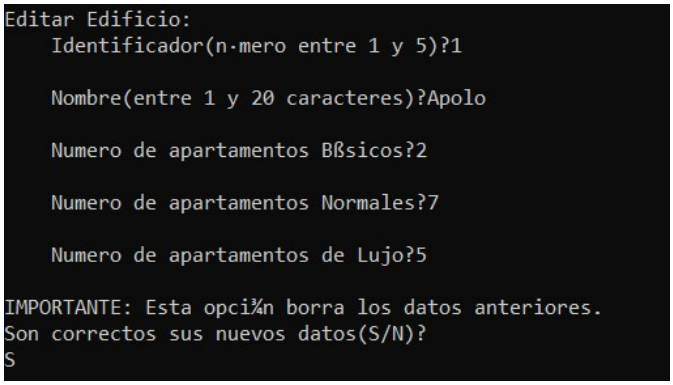

Figure 3: Menú Edición de Edificios

Pulsando 'L' se accede a la Lista de Edificios. Se muestra en pantalla cada edificio con sus datos correspondientes, aquellos que no tienen ningún dato aparecen como edificio dado de baja.

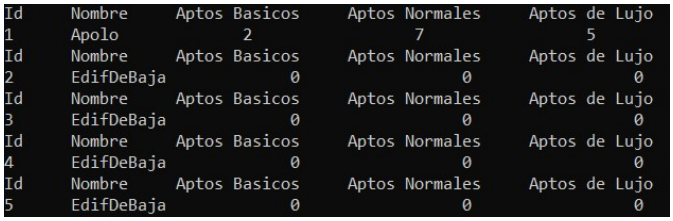

Figure 4: Listado de Edificios

Para ver los Apartamentos Disponibles en una fecha concreta, se pulsa la tecla 'A'. El programa pide el Identificador del Edificio, la Fecha y la Duración de la estancia a comprobar.

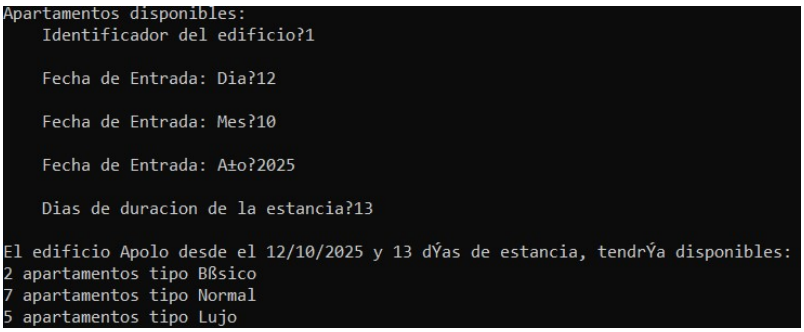

Figure 5: Menú Apartamentos Disponibles

Para reservar un apartamento se pulsa la tecla 'R'. Se introduce el Identificador del edificio, el Tipo de Apartamento deseado, la Fecha y la Duracion de la estancia. El programa imprime por pantalla los datos de la reserva, junto con la referencia del apartamento a reservar.

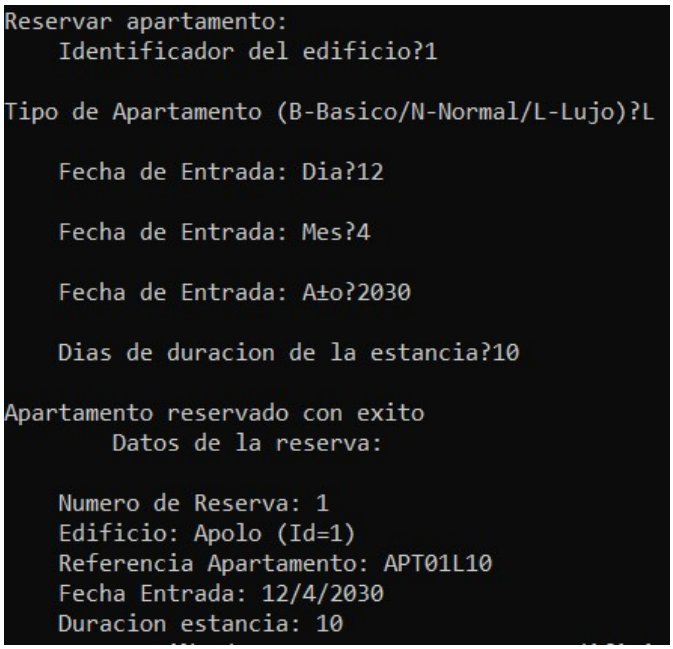

Figure 6: Menú Reservas de Apartamentos

6 Materiales empleados

- Ide para implementar la práctica: C $\pm$
- Procesador de texto para la memoria: LaTeX