```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include <time.h>
#include <conio.c>
#include <ctype.h>

#define MAXID   5
#define INCEST  5
#define EMPINI 2
#define LENNOM 41
#define LENAPE 31
#define MAXEMP 10
#define ARRIBA 72
#define ABAJO 80
#define ENTER  13
#define CANTOPC 5
#define TAMAOPC 30
#define MAXCAR 99

#define NOMARCH "C:\\TEMP\\DATOSEMP.DAT"


typedef struct
{
    int dia,mes,agno;
}FECHA;

typedef struct
{
    char idemp[MAXID];
    char nombre1[LENNOM];
    char nombre2[LENNOM];
    char apellido1[LENAPE];
    char apellido2[LENAPE];
    FECHA fnacim;
    FECHA fentrada;
    float salario;
    char StActivo; // ‘T’ activo y ‘F’inactivo
}EMP;

EMP capemp(int);
int valfecha(EMP );
int edadhoy(EMP );
int actual();
int actualmes();
int actualdia();
int movimiento(int);
void mostrar(int);
void colordefault(void);
void ponercolor(int, int);
void listanombre (int, EMP[]);
void menu1();
void menu2();
void impemp(EMP, int ,int);
void buscarnom( int, EMP[]);
void buscarape( int, EMP[]);
void buscareda( int, EMP[]);
void buscarestatus( int, EMP[]);

int main()
{
    EMP *empl;
    int ind=0, totemp, salir=0, modi=0, pos=0, opc=0;
    char input[2], input2[2], caract;
```

```
67      long tamarchiv;
68      FILE *ArchEmp;
69
70
71
72      do
73      {
74
75          system("cls");
76          menu1();
77          fflush(stdin);
78          input[0] = getchar();
79          fflush(stdin);
80
81          switch(input[0])
82          {
83
84              case '1':
85              {
86
87
88                  ArchEmp = fopen(NOMARCH, "rt");
89
90                  if(ArchEmp == NULL)
91                  {
92                      totemp = EMPINI;
93                      empl = (EMP *) malloc(totemp * sizeof(EMP));
94                  }
95                  else
96                  {
97
98                      fseek(ArchEmp, 0L, SEEK_END);
99                      tamarchiv = ftell(ArchEmp);
100                     totemp = tamarchiv / sizeof(EMP);
101
102                     ind = totemp;
103                     totemp += EMPINI;
104                     empl = (EMP *) malloc(totemp * sizeof(EMP));
105                     rewind(ArchEmp);
106
107                     fread(empl, totemp, 1, ArchEmp );
108                     fclose(ArchEmp);
109
110
111                 }
112                 printf("Ingrese los datos de el nuevo usuario\n\n");
113                 // printf("Cantidad de empleados a capturar : ");
114
115                 do
116                 {
117                     if( ind == totemp)
118                     {
119                         totemp += INCEST;
120                         printf("\nReasignando memoria...\n");
121                         empl = (EMP *) realloc(empl, totemp * sizeof(EMP));
122                     }
123
124                     printf("\nEmpleado %02d: \n",ind+1);
125                     empl[ind++] = capemp(ind);
126
127                     printf("%cDesea capturar otro empleado? (Si) o (No) \n\n",168);
128
129                     do
130                     {
131                         caract = (tolower(getch()));
132
```

```
133                     }while(caract != 's' && caract != 'n');
134
135
136                 }while(caract == 's');
137
138                 totemp = ind;
139
140                 ArchEmp  = fopen(NOMARCH, "a");
141
142                 if( ArchEmp == NULL)
143                 {
144                     printf("Error creando archivo de salida...\n");
145                     exit(-1);
146                 }
147
148                 fwrite(empl, totemp * sizeof(EMP), 1, ArchEmp);
149                 fclose(ArchEmp);
150
151                 system("cls");
152                 break;
153             }
154
155             case '2':
156             {
157
158                 do
159                 {
160
161                     system("cls");
162                     menu2();
163
164                     fflush(stdin);
165                     input2[0] = tolower(getchar());
166                     fflush(stdin);
167
168                     switch(input2[0])
169                     {
170                         case 'n':
171                         {
172                             salir = 0;
173                             printf("Ingrese un nombre para buscar: ");
174
175                             buscarnom( totemp, empl);
176
177                             getch();
178                             break;
179                         }
180
181                         case 'a':
182                         {
183                             salir = 0;
184                             printf("Ingrese el apellido por el cual se buscara: ");
185
186                             buscarape( totemp, empl);
187
188                             getch();
189                             break;
190                         }
191
192                         case 'e':
193                         {
194                             salir = 0;
195                             printf("Ingrese la edad a buscar: \n");
196
197                             buscareda( totemp, empl);
198                             getch();
```

```
                            break;
                        }

                        case 'v':
                        {
                            salir = 1;
                            system("cls");
                            break;
                        }

                        case 't':
                        {
                            printf("Digite (s) o (n) para buscar empleados por
estatus: \n");

                            buscarestatus(totemp, empl);
                            getch();

                            break;
                        }

                        case 's':
                        {
                            return 0;
                            break;
                        }

                        default:

                        puts("Opcion no valida\n");
                        getch();
                    }


                }while(input2[0] != 's' && salir != 1);


                break;

            }

            case '3':
            {
                do
                {


                    system("cls");
                    gotoxy(1,1);
                    puts("\t\t\t Bienvenido al menu de listar.\n\n");
                    puts("\t Seleccione con ENTER y las flachas de direccion ARRIBA
y ABAJO.\n\n");
                    salir = 0;
                    opc = movimiento(CANTOPC);


                    system("cls");

                    switch(opc)
                    {

                        case 0:
                        {
                            colordefault();
                            puts("Digite el patron por el que desea buscar: ");

```

```
                        listanombre(totemp, empl);
                        getch();
                        break;
                    }

                    case 1:
                    {
                        colordefault();
                        puts("Menu de salario\n");
                        getch();
                        break;
                    }

                    case 2:
                    {
                        colordefault();
                        puts("Menu de edades\n");
                        getch();
                        break;
                    }

                    case 3:
                    {
                        salir = 1;
                        break;
                    }

                    case 4:
                    {
                        return 0;
                        break;
                    }


                }

            }while( opc != 5 && salir != 1);

            printf("Esta es la lista de los usuarios registrados\n");

            ArchEmp  = fopen(NOMARCH, "rb");
            //totemp = ind;

            if( ArchEmp == NULL)
            {
                printf("Error creando archivo de salida...\n");
                exit(-1);
            }
            else
            {
                fseek(ArchEmp, 0L, SEEK_END);
                tamarchiv = ftell(ArchEmp);
                totemp = tamarchiv / sizeof(EMP);

                //ind = totemp;
                //totemp += EMPINI;
                empl = (EMP *) malloc(totemp * sizeof(EMP));
                rewind(ArchEmp);

                fread(empl, totemp, 1, ArchEmp );
                fclose(ArchEmp);

                //Imprimir empleados
                printf("Empleados capturados...\n\n");
                for(ind = 0; ind < totemp; ind++ )
                {
```

```
                        printf("\t \tEmpleado %02d \n ",ind+1);
                        impemp(empl[ind],1, ind+1);
                    }
                }

                fclose(ArchEmp);*/



                break;
            }

            case '4':
            {

                system("cls");
                printf("\n Empleados capturados...\n\n");

                for(ind = 0; ind < totemp; ind++ )
                {
                    printf("\t \tEmpleado %02d \n ",ind+1);
                    impemp(empl[ind],1, ind+1);
                }

                do
                {
                    printf("Digite el id del empleado que desea modificar: ");
                    fflush(stdin);
                    scanf("%d", &modi);
                    printf("\n%d", modi);
                    fflush(stdin);


                }while( modi <= ind);

                printf("\nEmpleado %02d: \n",modi);

                for(ind = 0 ; ind < totemp; ind++)
                {

                    if( ind == modi)
                    {
                        pos = ind;
                        empl[pos] = capemp(modi);
                    }

                }

                getch();
                break;
            }

            case '5':
            {
                return 0;
                break;
            }

            default:

            puts("\nOpcion no valida.  \n");
            getch();

        }

    }while(input[0] != '5');
```

```c
    free(empl);
    return 0;
}

/*
Funcion: menu1
Argumentos: Nada.
Objetivo: Imprimir las opciones del menu1.
Retorno: Ningun valor.
*/

void menu1()
{
    puts("\t\t Bienvenido al programa\n\n");
    puts("Digite 1: para ingresar un nuevo empleado.");
    puts("Digite 2: para buscar un empleando.");
    puts("Digite 3: para listar los empleados presentes.");
    puts("Digite 4: para modificar un empleado.");
    puts("Digite 5: para salir.");
}

/*
Funcion: menu2
Argumentos: Nada.
Objetivo: Imprimir las opciones del menu2.
Retorno: Ningun valor.
*/

void menu2()
{
    printf("\t\t Bienvenido al menu de busqueda \n\n");
    puts("Digite n: para buscar por nombre.");
    puts("Digite a: para buscar por apellido.");
    puts("Digite e: para buscar por edad.");
    puts("Digite t: para buscar por estatus. ");
    puts("Digite v: para volver al menu anterior. ");
    puts("Digite s: para salir. ");

}

/*
Funcion: capemp
Argumentos: int c: Genera el id.
Objetivo: Capturar los datos de los empleados.
Retorno: Ningun valor.
*/

EMP capemp(int c)
{
    EMP emp;

    int ind, f=0, maxed=0, fen=0, fna=0, ap=0, nom=0;


    printf("Id empleado: ");
    fflush(stdin);
    printf("%04d\n",c);

    do
    {
        f=0;
        nom = 0;
        printf("Primer nombre: ");
        fflush(stdin);
```

```
461           memset( emp.nombre1, 0, sizeof(emp.nombre1) );
462           gets(emp.nombre1);
463           fflush(stdin);
464
465           //Comprobar de que el nombre no este en blanco.
466           if(emp.nombre1[0] == '\0' || emp.nombre1[0] == 0 || strcmp(emp.nombre1, "")
== 0 ||  strlen(emp.nombre1) == 0)
467           {
468               nom=1;
469           }
470
471           for( ind =0; ind<strlen(emp.nombre1); ind++)
472           {
473               //Transformar a mayusculas
474               if (emp.nombre1[ind] >= 'a' && emp.nombre1[ind] <= 'z'  )
475               {
476                   emp.nombre1[ind] -= 32;
477               }
478
479               if( emp.nombre1[ind] != 'ñ' || emp.nombre1[ind] != 'Ñ')
480               {
481                   //Revisar si es guion o espacio
482                   if (emp.nombre1[ind] != 45 && emp.nombre1[ind] != 32 )
483                   {
484
485                       if( emp.nombre1[ind] < 65 || emp.nombre1[ind] > 90 )
486                       {
487                           f=1;
488                           ind=strlen(emp.nombre1);
489                       }
490
491                   }
492               }
493
494
495
496           }
497
498           if(f == 1)
499           {
500               printf("Solo se permiten espacios, letras o guiones.\n");
501           }
502
503           if(nom == 1)
504           {
505               printf("El nombre no puede estar en blanco.\n");
506           }
507
508
509
510       }while(f ==1 || nom == 1 );
511
512
513
514       do
515       {
516           f=0;
517           printf("Segundo nombre: ");
518           fflush(stdin);
519           memset( emp.nombre2, 0, sizeof(emp.nombre2) );
520
521           gets(emp.nombre2);
522           fflush(stdin);
523
524
525           for( ind =0; ind<strlen(emp.nombre2); ind++)
```

```
526          {
527              if (emp.nombre2[ind] >= 'a' && emp.nombre2[ind] <= 'z' )
528              {
529                  emp.nombre2[ind] -= 32;
530              }
531
532
533              if (emp.nombre2[ind] != 45 && emp.nombre2[ind] != 32 )
534              {
535
536                  if(emp.nombre2[ind] < 65 || emp.nombre2[ind] > 90)
537                  {
538                      f=1;
539                      ind=strlen(emp.nombre2);
540
541                  }
542
543              }
544
545          }
546
547          if(f==1)
548          {
549              printf("Solo se permiten espacios, letras o guiones.\n");
550          }
551
552
553      }while(f ==1);
554
555
556      do
557      {
558          f=0;
559          ap = 0;
560          printf("Primer Apellido: ");
561          fflush(stdin);
562          memset( emp.apellido1, 0, sizeof(emp.apellido1) );
563          gets(emp.apellido1);
564          fflush(stdin);
565
566          if(emp.apellido1[0] == '\0' || emp.apellido1[0] == 0 || strcmp(emp.
apellido1, "") == 0 ||  strlen(emp.apellido1) == 0)
567          {
568              ap=1;
569          }
570
571
572          for( ind =0; ind<strlen(emp.apellido1); ind++)
573          {
574              if (emp.apellido1[ind] >= 'a' && emp.apellido1[ind] <= 'z' )
575              {
576                  emp.apellido1[ind] -= 32;
577              }
578
579
580              if (emp.apellido1[ind] != 45 && emp.apellido1[ind] != 32 )
581              {
582                  if( emp.apellido1[ind] < 65 || emp.apellido1[ind] > 90 )
583                  {
584                      f=1;
585                      ind=strlen(emp.apellido1);
586
587                  }
588
589              }
590
```

```
591         }
592
593         if(f==1)
594         {
595             printf("Solo se permiten espacios, letras o guiones.\n");
596         }
597
598          if(ap == 1)
599         {
600             printf("El apellido no puede estar en blanco.\n");
601         }
602
603
604
605
606     }while( f ==1 || ap == 1);
607
608
609     do
610     {
611         f=0;
612         printf("Segundo Apellido: ");
613         fflush(stdin);
614         memset( emp.apellido2, 0, sizeof(emp.apellido2) );
615         gets(emp.apellido2);
616         fflush(stdin);
617
618         for( ind =0; ind<strlen(emp.apellido2); ind++)
619         {
620             if (emp.apellido2[ind] >= 'a' && emp.apellido2[ind] <= 'z' )
621             {
622                 emp.apellido2[ind] -= 32;
623             }
624
625
626             if (emp.apellido2[ind] != 45 && emp.apellido2[ind] != 32 )
627             {
628
629                 if(emp.apellido2[ind] < 65 || emp.apellido2[ind] > 90)
630                 {
631                     f=1;
632                     ind=strlen(emp.apellido2);
633
634                 }
635
636             }
637
638         }
639
640         if(f==1)
641         {
642             printf("Solo se permiten espacios, letras o guiones.\n");
643         }
644
645     }while(f == 1);
646
647
648     printf("Sueldo: ");
649     fflush(stdin);
650     scanf("%f",&emp.salario);
651     fflush(stdin);
652
653
654
655     do
656     {
```

```
 657          printf("Fecha de nacimiento(dd/mm/yyyy): ");
 658          fflush(stdin);
 659          scanf("%d/%d/%d",&emp.fnacim.dia,&emp.fnacim.mes,&emp.fnacim.agno);
 660          fflush(stdin);
 661
 662          if(emp.fnacim.mes > 12 || emp.fnacim.mes <= 0 || emp.fnacim.agno <= 1850 ||
emp.fnacim.dia <= 0 || emp.fnacim.dia > 31 || emp.fnacim.agno > actual() )
 663          {
 664              if((emp.fnacim.dia <= 0 || emp.fnacim.dia > 31) && emp.fnacim.mes != 2)
 665              {
 666                  printf("El dia debe ser entre el 1 y %d. \n", valfecha(emp));
 667              }
 668              if( emp.fnacim.agno < 1900)
 669              {
 670                  printf("El a%co debe ser mayor a 1900.\n", 164);
 671              }
 672
 673              if( emp.fnacim.agno > actual())
 674              {
 675                  printf("La fecha de nacimiento esta incorrecta. \n");
 676              }
 677
 678              if(emp.fnacim.mes == 2)
 679              {
 680
 681                  if(valfecha(emp) == 29)
 682                  {
 683                      printf("El dia debe ser entre el 1 y el 29 porque el a%co es
bisiesto. \n", 164);
 684                  }
 685                  else
 686                  {
 687                      printf("El dia debe ser entre el 1 y el 28. \n");
 688                  }
 689
 690              }
 691              if( emp.fnacim.mes > 12)
 692              {
 693                  printf("La cantidad de meses maximo es 12.\n");
 694              }
 695
 696          }
 697
 698      }while(emp.fnacim.mes > 12 || emp.fnacim.mes <= 0 || emp.fnacim.agno <= 1900 ||
emp.fnacim.dia <= 0 || emp.fnacim.dia > 31 || emp.fnacim.agno > actual());
 699
 700
 701      do
 702      {
 703          printf("Fecha de ingreso(dd/mm/yyyy): ");
 704          fflush(stdin);
 705          scanf("%d/%d/%d",&emp.fentrada.dia,&emp.fentrada.mes,&emp.fentrada.agno);
 706          fflush(stdin);
 707
 708          fen = emp.fentrada.agno;
 709          fna = emp.fnacim.agno;
 710          maxed = fen - fna;
 711
 712
 713          if(emp.fentrada.mes > 12 || emp.fentrada.mes <= 0 || emp.fentrada.agno <=
1900 || emp.fentrada.dia <= 0  || maxed < 18 || emp.fentrada.dia > valfecha(emp) || (
actual()-fna) > 116 || emp.fentrada.agno > actual())
 714          {
 715              if(emp.fentrada.dia <= 0 && emp.fentrada.mes != 2)
 716              {
 717                  printf("El dia debe ser entre el 1 y %d. \n", valfecha(emp));
```

```
 718             }
 719
 720             if(emp.fentrada.dia > valfecha(emp))
 721             {
 722                 printf("El mes que ha entrado no llega a esa cantidad de dias. \n"
);
 723             }
 724
 725             if( emp.fentrada.agno < 1900)
 726             {
 727                 printf("El a%cdo debe ser mayor a 1900.\n", 164);
 728             }
 729
 730             if((actual()-fna) > 116)
 731             {
 732                 printf("Lo sentimos no se admiten fantasmas (No existe nadie con
esta edad). \n");
 733             }
 734
 735             if( maxed < 18 )
 736             {
 737                 printf("La fecha de entrada debe ser mayor a la fecha de nacimiento
en 18 a%cos. \n", 164);
 738                 fen = 0;
 739                 fna = 0;
 740                 maxed = 0;
 741             }
 742
 743             if( emp.fentrada.agno > actual())
 744             {
 745                 printf("La fecha de entrada esta incorrecta. \n");
 746             }
 747
 748             if(emp.fentrada.mes > 12)
 749             {
 750                 printf("La cantidad de meses maximo es 12.\n");
 751             }
 752
 753         }
 754
 755     }while(emp.fentrada.mes > 12 || emp.fentrada.mes <= 0 || emp.fentrada.agno <=
1850 || emp.fentrada.dia <= 0 || emp.fentrada.dia > 31 || maxed < 18 || emp.fentrada.dia
> valfecha(emp) || (actual()-fna) > 116 || emp.fentrada.agno > actual());
 756
 757
 758     do
 759     {
 760         printf("Digite si el empleado esta activo(S)i o (N)o: ");
 761         fflush(stdin);
 762         emp.StActivo = (toupper(getch()));
 763         fflush(stdin);
 764
 765         if((emp.StActivo != 'S' && emp.StActivo != 'N') )
 766         {
 767             printf("\nError! Digite de nuevo.\n ");
 768         }
 769
 770     }while( emp.StActivo != 'S' && emp.StActivo != 'N');
 771
 772     printf("\n");
 773
 774     return emp;
 775
 776 }
 777
 778 /*
```

```
779  Funcion: impemp
780  Argumentos: EMP emp: Estructura con los datos de los empleados.
781  int titulo: Imprimir el encabezado.
782  int c: Obtener el valor del id.
783  Objetivo: Imprimir los datos de los empleados.
784  Retorno: Ningun valor.
785  */
786
787  void impemp(EMP emp, int titulo, int c)
788  {
789      int n1=0,n2=0, ap1=0, ap2=0, eda=0, esp=0;
790
791      n1=strlen(emp.nombre1);
792      n2=strlen(emp.nombre2);
793      ap1=strlen(emp.apellido1);
794      ap2=strlen(emp.apellido2);
795      eda = edadhoy(emp);
796      esp = 5;
797
798      if ( titulo )
799      {
800          printf("%-*s %-*s  %-*s  %s %s %s \n\n" , n1+n2, "Id Emp", ap1+ap2+n2+1,
"Nombre", 2,"Sueldo", "Fecha Ingreso", "Edad", "Activo(S)i o (N)o" );
801          printf("%04d %-*s %-*s %-*s %-*s %.2f %02d/%02d/%04d %d %-* %c\n ",
802          c,n1+n2,emp.nombre1,n2,emp.nombre2,ap1,emp.apellido1,ap2,emp.apellido2,emp.
salario,
803          emp.fentrada.dia,emp.fentrada.mes,emp.fentrada.agno, eda, esp, emp.StActivo
);
804
805          printf("\n\n\n");
806      }
807
808
809      return;
810  }
811
812  /*
813  Funcion: valfecha
814  Argumentos: EMP emp: Estructura con los datos de los empleados.
815  Objetivo: Saber la cantidad de dias del mes.
816  Retorno: Dias del mes.
817  */
818
819  int valfecha(EMP emp)
820  {
821
822
823      int numberOfDays, m, a, es=0;
824
825      m = emp.fentrada.mes;
826      a= emp.fentrada.agno;
827
828      if (m == 4 ||m == 6 || m == 9 || m == 11)
829          numberOfDays = 30;
830
831      else if (m == 2)
832      {
833        es = (a % 4 == 0 && a % 100 != 0) || ( a % 400 == 0);
834
835          if(es)
836          {
837              numberOfDays=29;
838          }
839          else
840          {
841              numberOfDays=28;
```

```
842         }
843 
844     }
845     else
846       numberOfDays = 31;
847 
848     return numberOfDays;
849 
850 }
851 
852 /*
853 Funcion: actual
854 Argumentos: Nada.
855 Objetivo: Obtener el año actual.
856 Retorno: Año actual.
857 */
858 int actual()
859 {
860 
861     time_t now = time(NULL);
862     struct tm *t = localtime(&now);
863 
864     return t->tm_year+1900;
865 
866 }
867 
868 /*
869 Funcion: actualmes
870 Argumentos: Nada.
871 Objetivo: Obtener el mes actual.
872 Retorno: Mes actual.
873 */
874 int actualmes()
875 {
876 
877     time_t now = time(NULL);
878     struct tm *t = localtime(&now);
879 
880     return t->tm_mon+1;
881 
882 }
883 
884 /*
885 Funcion: actualdia
886 Argumentos: Nada.
887 Objetivo: Obtener el dia actual.
888 Retorno: Dia actual.
889 */
890 
891 int actualdia()
892 {
893 
894     time_t now = time(NULL);
895     struct tm *t = localtime(&now);
896 
897 
898     return t->tm_mday;
899 
900 }
901 
902 /*
903 Funcion: buscarnom
904 Argumentos: int totemp: total empleados.
905             EMP empl[]: Estructura de empleados
906 Objetivo: Buscar por nombre.
907 Retorno: Ningun valor.
```

```
 908  */
 909
 910  void buscarnom( int totemp, EMP empl[])
 911  {
 912      char buscarnombre[LENNOM];
 913      int ind;
 914
 915      fflush(stdin);
 916      gets(buscarnombre);
 917      fflush(stdin);
 918
 919      for(ind =0; ind < totemp; ind++)
 920      {
 921
 922          if( (strcmp(toupper(buscarnombre), empl[ind].nombre1) == 0 ) || strcmp(
toupper(buscarnombre), empl[ind].nombre2) == 0 )
 923          {
 924              impemp(empl[ind], 1, ind+1);
 925          }
 926
 927      }
 928
 929
 930  }
 931
 932  /*
 933  Funcion: buscarape
 934  Argumentos: int totemp: total empleados.
 935              EMP empl: Estructura de empleados
 936  Objetivo: Buscar por apellido.
 937  Retorno: Ningun valor.
 938  */
 939  void buscarape(int totemp, EMP empl[])
 940  {
 941      int ind;
 942      char buscarapellido[LENAPE];
 943
 944      fflush(stdin);
 945      gets(buscarapellido);
 946      fflush(stdin);
 947
 948      for( ind =0; ind < totemp; ind++)
 949      {
 950          if((strcmp( toupper(buscarapellido), empl[ind].apellido1) == 0) || (strcmp(
toupper(buscarapellido), empl[ind].apellido2) == 0))
 951          {
 952
 953              impemp(empl[ind], 1, ind+1);
 954
 955          }
 956
 957      }
 958
 959
 960  }
 961
 962  /*
 963  Funcion: buscareda
 964  Argumentos: int totemp: total empleados.
 965              EMP empl: Estructura de empleados
 966  Objetivo: Buscar por edad.
 967  Retorno: Ningun valor.
 968  */
 969
 970  void buscareda( int totemp, EMP empl[])
 971  {
```

```
    int ind = 0, digitedad = 0;



    fflush(stdin);
    scanf("%d", &digitedad);
    fflush(stdin);

    for( ind =0; ind < totemp; ind++)
    {

        if(digitedad == edadhoy(empl[ind]))
        {
            impemp(empl[ind], 1, ind+1);
        }


    }


}

/*
Funcion: buscarestatus
Argumentos: int totemp: total empleados.
            EMP empl: Estructura de empleados
Objetivo: Buscar por estatus.
Retorno: Ningun valor.
*/

void buscarestatus( int totemp, EMP empl[])
{
    int ind;
    char estatu;

    fflush(stdin);
    estatu = getchar();
    fflush(stdin);


    for( ind = 0; ind < totemp; ind++)
    {
        if(toupper(estatu)== empl[ind].StActivo)
        {
            impemp(empl[ind], 1, ind+1);
        }
    }

}

/*
Funcion: edadhoy
Argumentos: EMP empl: Estructura de empleados
Objetivo: Edad del empleado.
Retorno: Edad del empleado.
*/

int edadhoy(EMP empl)
{
    struct tm *tiempo;
    int dia=0 ,mes=0 ,anio=0 , edad=0;

    time_t fecha_sistema;
    time(&fecha_sistema);
    tiempo = localtime(&fecha_sistema);

```

```
1038      anio = tiempo->tm_year + 1900;
1039      mes = tiempo->tm_mon + 1;
1040      dia = tiempo->tm_mday;
1041
1042      edad = anio - empl.fnacim.agno;
1043
1044      if(mes > empl.fnacim.mes)
1045      {
1046         edad++;
1047      }
1048
1049      if (mes == empl.fnacim.mes)
1050      {
1051          if(dia > empl.fnacim.dia)
1052          {
1053              edad++;
1054          }
1055      }
1056
1057      return edad;
1058  }
1059
1060  /*
1061  Funcion: movimiento
1062  Argumentos: int n: total de empleados
1063  Objetivo: Moverse.
1064  Retorno: Nada.
1065  */
1066
1067  int movimiento( int n)
1068  {
1069      int ind, sel=0;
1070      _setcursortype(0);
1071      char tecla;
1072      //system("cls");
1073      do
1074      {
1075          gotoxy(1, 5);
1076          mostrar(sel);
1077
1078          tecla = getch();
1079
1080          if( tecla == ABAJO)
1081          {
1082              if(sel == n - 1)
1083              {
1084                  sel = 0;
1085              }
1086              else
1087              {
1088                  sel++;
1089              }
1090          }
1091
1092          if( tecla == ARRIBA)
1093          {
1094              if( sel == 0)
1095              {
1096                  sel = n -1;
1097              }
1098              else
1099              {
1100                  sel--;
1101              }
1102          }
1103
```

```
1104
1105        }while( tecla != ENTER);
1106
1107        _setcursortype(1);
1108        colordefault();
1109        return sel;
1110
1111    }
1112
1113    /*
1114    Funcion: mostrar
1115    Argumentos: int sel: seleccion para el color
1116    Objetivo: Seleccionar la opcion de un color e imprimirla.
1117    Retorno: Nada.
1118    */
1119
1120    void mostrar(int sel)
1121    {
1122
1123        int ind;
1124        char menu3[CANTOPC][TAMAOPC] = {"Nombres con un patron.",
1125                                        "Salario en un rango   ",
1126                                        "Edad en un rango.     ",
1127                                        "Menu anterior.        ",
1128                                        "Salir.                "};
1129
1130        for( ind =0; ind < CANTOPC; ind++)
1131        {
1132            if( sel == ind)
1133            {
1134                ponercolor(YELLOW, BLUE);
1135            }
1136            else
1137            {
1138                ponercolor(BLUE, LIGHTGRAY);
1139            }
1140            printf( "%s \n", menu3[ind]);
1141        }
1142
1143
1144    }
1145
1146    /*
1147    Funcion: ponercolor
1148    Argumentos: int tc: color del texto
1149                int tbc: color del background
1150    Objetivo: poner color al texto y background
1151    Retorno: Nada.
1152    */
1153
1154    void ponercolor(int tc, int tbc)
1155    {
1156        textbackground(tbc);
1157        textcolor(tc);
1158    }
1159
1160    /*
1161    Funcion: colordefault
1162    Argumentos: Nada
1163    Objetivo: poner color al texto y background
1164    Retorno: Nada.
1165    */
1166
1167    void colordefault()
1168    {
1169        ponercolor(LIGHTGRAY, BLACK);
```

```
1170
1171  }
1172
1173  /*
1174  Funcion: listanombre
1175  Argumentos: int totemp: Total de empleados
1176              EMP empl[]: estructura con los datos de los empleados
1177  Objetivo: imprimir los nombres por un patron
1178  Retorno: Nada.
1179  */
1180
1181  void listanombre (int totemp, EMP empl[])
1182  {
1183      int ind, ind2;
1184      char patron[5];
1185
1186      fflush(stdin);
1187      gets(patron);
1188      fflush(stdin);
1189
1190      for( ind = 0; ind < totemp; ind++)
1191      {
1192          for(ind2 = 0; ind2 < strlen(patron); ind2++)
1193          {
1194              if( strcmp(toupper(patron), empl[ind].nombre1[ind2]) ==0 || strcmp(
toupper(patron), empl[ind].nombre2[ind2]) ==0 || strcmp(toupper(patron), empl[ind].
apellido1[ind2]) ==0 || strcmp(toupper(patron), empl[ind].apellido2[ind2]) ==0)
1195              {
1196                  impemp(empl[ind],1, ind+1);
1197              }
1198          }
1199      }
1200
1201  }
1202
```