Anexo C. Ejercicios nivel 2 (medio-difícil)

Introducción

En este anexo incluimos una colección de ejercicios que podríamos considerar de un nivel medio/difícil. Para solucionarlos se deben conocer, además de los tipos de datos simples y las estructuras de control, la programación modular y los tipos de datos avanzados. Las soluciones de los ejercicios vienen al final del capítulo, pero no las consultes hasta que no los intentes solucionar por ti mismo.

Ejercicios

A continuación presentamos los enunciados de los ejercicios propuestos, ordenados de menor a mayor dificultad.

Ejercicio 1

En este ejercicio vamos a implementar un pequeño juego en el que solicitaremos al usuario una palabra secreta. A continuación, un segundo usuario debe adivinar la palabra en tres intentos como máximo. Para ayudarle, en cada intento el programa mostrará la palabra secreta ocultando con asteriscos el 75% de sus caracteres (en posiciones aleatorias y distintas en cada ocasión). Si se superan los tres intentos sin acertar la palabra, el programa la mostrará y finalizará. En caso de acertar la palabra, felicitaremos al jugador y terminaremos.

Ejercicio 2

Realizar un programa completo en C que solicite al usuario la edad de N personas de una población (N es una constante del programa). A continuación mostrará por pantalla la edad máxima, la mínima y la edad media de la población. Además, nos dirá el número de personas de cada edad distinta en la población. Supondremos que todas las edades se encuentran entre 0 y 100 años.

Ejercicio 3

Queremos realizar un programa para gestionar las calificaciones medias de un grupo de alumnos de la UA. Todos los alumnos del grupo tienen las mismas asignaturas, y nos interesa saber la nota media de cada asignatura, la nota media de cada alumno y la nota media del curso. Al comenzar, el programa solicitará para cada uno de los NALU alumnos, la nota numérica de las NASI asignaturas. A continuación mostrará la nota media de cada alumno, la nota media de cada asignatura y la nota media del curso en general.

Ejercicio 4

Realizar un programa completo en C que realice el siguiente cálculo: V1 x M x V2, donde V1 es un vector de NFIL elementos, M es una matriz de NFILxNCOL y V2 es un vector de tamaño NCOL (NFIL y NCOL serán constantes del programa). Tanto la matriz como los vectores se inicializarán con valores enteros aleatorios entre 0 y 100. El programa mostrará los vectores y la matriz originales, y el resultado de la operación (el resultado final de dicha operación consta de un solo número).

Ejercicio 5

Escribir un programa completo en C que realice el siguiente juego: al comenzar, el programa creará una matriz cuadrada de NxN elementos de tipo entero (N es una constante del programa) que rellenará con valores aleatorios entre 0 y N^2 (si N vale 4, por ejemplo, rellenará con valores entre 0 y 15). La idea es que el usuario adivine los números que componen el tablero en menos de 10 oportunidades. Al principio, ocultaremos todas las casillas del tablero con asteriscos y en cada iteración solicitaremos un nuevo valor al usuario. Si acierta, descubriremos las casillas con el valor indicado y si no, incrementaremos el número de fallos. La única pista que tendrá el usuario será la suma de los elementos de la matriz por filas y por columnas. El programa finalizará cuando el usuario logre descubrir todas las casillas o bien cuando llegue a 10 intentos fallidos.

Ejercicio 6

En este ejercicio vamos a implementar el juego del buscaminas simplificado. Para ello, mostraremos al usuario un tablero de NxN casillas ocultas (signo '?'), en las que el programa esconderá aleatoriamente 10 minas. En cada ocasión, aparecerá un menú en el que podremos: 1) tirar 2) hacer trampa 3) terminar. Si pulsamos 1, nos pedirá fila y columna. Si en esa posición había una mina, se descubrirá el tablero completo y el juego acabará (perdiendo el usuario, claro). Si esa posición estaba en blanco, colocará una 'X' y seguiremos jugando. En caso de completar todo el tablero sin encontrar ninguna mina, habremos finalizado el juego (ganando, evidentemente). Si el usuario pulsa 2, le mostraremos el tablero con las minas y continuará jugando. En el caso de pulsar 3, el programa terminará.

Ejercicio 7

En este ejercicio vamos a implementar el juego del Sudoku. El programa mostrará el panel de 9x9 con los números introducidos hasta el momento y con un hueco en blanco en las casillas vacías (al principio, todas las casillas estarán vacías). En cada ocasión, nos pedirá la fila y la columna donde tirar y el valor (de 1 a 9) que deseamos. Si ese valor es incorrecto (el mismo valor aparece en la misma fila o columna, o en el mismo cuadrante), el programa no nos dejará tirar. El programa terminará cuando el usuario consiga completar el juego.

Ejercicio 8

En este ejercicio vamos a implementar un juego basado en los típicos puzzles compuestos por una serie de piezas que forman una figura y un hueco que permite mover las 4 piezas que tiene alrededor. En nuestro caso, se tratará de una matriz cuadrada de NxN elementos (N será una constante del programa) compuesta por N^2-1 números y un hueco (por ejemplo, si N=4, será una matriz de 4x4 casillas 15 de ellas numeradas de 1 a 15 y un hueco). Para resolver el juego, es necesario que el usuario mueva el hueco hasta que los números de las casillas estén ordenados de izquierda a derecha y de arriba a abajo, y con el hueco en última posición. Para inicializar el tablero, se le preguntará al usuario el nivel de dificultad, que será entre 1 y 3. El procedimiento consistirá en rellenar el tablero con la solución (ordenado de izquierda a derecha y de arriba abajo, con el hueco en la última casilla) e ir haciendo tantos movimientos aleatorios del hueco como el nivel multiplicado por N^2. Por ejemplo, si se elige nivel de dificultad 3 y el tablero es de 4x4, el número de movimientos aleatorios del hueco será de 3x4x4 = 48. El juego terminará cuando el usuario abandone el juego con escape o de con el orden correcto del tablero.

Soluciones

A continuación presentamos las soluciones a los ejercicios propuestos. Recuerda que normalmente hay varias soluciones al mismo ejercicio. Aquí solo presentamos una solución válida, que no ha de ser la mejor. En los ejercicios de este nivel es importante realizar un buen diseño modular, así como utilizar convenientemente los tipos de datos avanzados (vectores y matrices). No consultes la solución hasta que no agotes las posibilidades, porque aprender a programar no consiste en comprender la solución sino en llegar a ella por ti mismo. Utiliza el compilador para probar tus soluciones.

Solución ejercicio 1

La solución a este ejercicio es muy sencilla. El único problema es cómo imprimir la cadena con un 75% de asteriscos de la forma más exacta posible. Para ello, los asteriscos se colocan sobre una cadena auxiliar, comprobando que no ponemos ningún asterisco encima de otro que ya hayamos puesto:

```
//librerías necesarias
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

//imprime la palabra con un 75% de asteriscos
void ImprimirPalabra(char oculta[]){
   char aux[100]; //cadena auxiliar
   strcpy(aux,oculta);
   int l=strlen(aux),i;
   int nast = l*75/100;
   while(nast>0){
      int n=rand()%l;
      if(aux[n]!='*'){
         aux[n]='*';
         nast--;
      }
   }
   printf("Palabra oculta: %s\n",aux);
   return;
}

void Juego(void){
   //variables necesarias
   char oculta[100], actual[100]; //palabra oculta
   int intentos=0;

   //solicitamos la palabra
   printf("Introduzca palabra oculta: ");
```

```
    gets(oculta);

    do{
        system("cls");
        printf("Intento %d\n",intentos+1);
        ImprimirPalabra(oculta);
        printf("Escriba la palabra: ");
        gets(actual);
    }while(intentos<3 && strcmp(oculta,actual)!=0);

    if(strcmp(oculta,actual)==0){
        printf("Felicidades! Has acertado la palabra\n");
    }else{
        printf("Lo siento, perdiste. La palabra era %s\n",oculta);
    }
    return;
}

//función main
int main(void) {
    Juego();
    getche();
    return(0);
}
```

Solución ejercicio 2

Una posible solución sería:

```
//librerías necesarias
#include <stdio.h>
#include <conio.h>
#include <string.h>

//Número de personas
#define N 10

//solicitud de las edades al usuario
void Solicitar(int edades[N]){
    int i;
    for(i=0;i<N;i++){
        //bucle para que siempre introduzca de 0 a 100
        do{
            printf("Introduzca la edad del ciudadano %d: ",i+1);
            scanf("%d",&edades[i]);
        }while(edades[i]<0 || edades[i]>100);
    }
    return;
}
```

```
//Cálculo de la edad máxima y de la mínima
void MinMax(int edades[N], int *minima, int *maxima){
   int i;
   *minima=*maxima=edades[0]; //inicializamos con la primera
   for(i=1;i<N;i++){
      if(edades[i]>*maxima) *maxima=edades[i];
      if(edades[i]<*minima) *minima=edades[i];
   }
   return;
}

//Cálculo de la edad media de la población
float Media(int edades[N]){
   float media=0;
   for(int i=0;i<N;i++) media+=edades[i];
   media/=N;
   return(media);
}

//número de personas de cada edad
void PersonasDeCadaEdad(int edades[N]){
   int i,j;
   for(i=0;i<=100;i++){ //para cada edad posible
      int npersonas=0; //buscamos cuántas personas la tienen
      for(j=0;j<N;j++){
         if(edades[j]==i){
            npersonas++;
         }
      }
      if(npersonas>0){
         printf("Numero de personas con %d anyos = %d\n",i,npersonas);
      }
   }
   return;
}

//función principal
void Problema(void){
   int edades[N];
   Solicitar(edades);
   int minima,maxima;
   MinMax(edades,&minima,&maxima);
   printf("Resultados: \n");
   printf("\tHabitante de menor edad: %d\n",minima);
   printf("\tHabitante de mayor edad: %d\n",maxima);
   float media=Media(edades);
   printf("\tEdad media de la poblacion: %f\n",media);
   PersonasDeCadaEdad(edades);
   return;
```

```
}

//función main
int main(void){
   Problema();
   getche();
   return(0);
}
```

Solución ejercicio 3

La solución más evidente consiste en utilizar una matriz de NASIxNALU, en la que almacenamos todas las calificaciones para después calcular las medias:

```
//librerías necesarias
#include <stdio.h>
#include <conio.h>
#include <string.h>

//Número de alumnos
#define NALU 5
//Número de asignaturas
#define NASI 3

//solicitud de las calificaciones de los alumnos
void SolicitarCalificaciones(float calif[NALU][NASI]){
   int i,j;
   for(i=0;i<NALU;i++){
      printf("Alumno %d\n",i+1);
      for(j=0;j<NASI;j++){
         //Solicitamos la nota entre 0 y 10
         do{
            printf("Calificacion de la asignatura %d: ",j+1);
            scanf("%f",&calif[i][j]);
         }while(calif[i][j]<0 || calif[i][j]>10);
      }
   }
   return;
}

//Media por alumno
void MediaAlumnos(float calif[NALU][NASI]){
   int i,j;
   for(i=0;i<NALU;i++){
      float media=0;
      for(j=0;j<NASI;j++){
         media+=calif[i][j];
      }
      media/=NASI;
      printf("Nota media del alumno %d: %f\n",i+1,media);
   }
```

```
    return;
}

//Media por asignatura
void MediaAsignaturas(float calif[NALU][NASI]){
    int i,j;
    for(j=0;j<NASI;j++){
        float media=0;
        for(i=0;i<NALU;i++){
            media+=calif[i][j];
        }
        media/=NALU;
        printf("Nota media de la asignatura %d: %f\n",j+1,media);
    }
    return;
}

//Media general
void MediaTotal(float calif[NALU][NASI]){
    int i,j;
    float media=0;
    for(j=0;j<NASI;j++){
        for(i=0;i<NALU;i++){
            media+=calif[i][j];
        }
    }
    media/=(NALU*NASI);
    printf("Nota media del curso: %f\n",media);
    return;
}

//función principal
void Problema(void){
    float calif[NALU][NASI];
    SolicitarCalificaciones(calif);
    MediaAlumnos(calif);
    MediaAsignaturas(calif);
    MediaTotal(calif);
    return;
}

//función main
int main(void){
    Problema();
    getche();
    return(0);
}
```

Solución ejercicio 4

Una posible solución sería:

```
//librerías necesarias
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//Número de filas y columnas
#define NFIL 4
#define NCOL 5

//Inicialización de la matriz
void InicializarMatriz(int M[NFIL][NCOL]){
   int i,j;
   for(i=0;i<NFIL;i++){
      for(j=0;j<NCOL;j++){
         M[i][j]=rand()%101;
      }
   }
   return;
}

//Inicialización de un vector
void InicializarVector(int V[], int tam){
   int i;
   for(i=0;i<tam;i++){
      V[i]=rand()%101;
   }
   return;
}

//Impresión de la matriz
void ImprimirMatriz(int M[NFIL][NCOL]){
   int i,j;
   for(i=0;i<NFIL;i++){
      for(j=0;j<NCOL;j++){
         printf("%d ",M[i][j]);
      }
      printf("\n");
   }
   return;
}

//Impresión de un vector
void ImprimirVector(int V[], int tam){
   int i;
   for(i=0;i<tam;i++){
      printf("%d ",V[i]);
   }
   printf("\n");
   return;
}
```

```
//Operación requerida
int Operacion(int V1[NFIL], int M[NFIL][NCOL], int V2[NCOL]){
   //primera operacion R = V1 x M
   int R[NCOL];
   int i,j;
   for(j=0;j<NCOL;j++){
      R[j]=0;
      for(i=0;i<NFIL;i++){
         R[j]+=V1[i]*M[i][j];
      }
   }
   //segunda operacion resul = R x V2
   int resul=0;
   for(j=0;j<NCOL;j++){
      resul+=R[j]*V2[j];
   }
   return(resul);
}

//función principal
void Problema(void){
   int V1[NFIL], M[NFIL][NCOL], V2[NCOL];
   InicializarVector(V1,NFIL);
   InicializarMatriz(M);
   InicializarVector(V2,NCOL);
   printf("V1:\n");
   ImprimirVector(V1,NFIL);
   printf("M:\n");
   ImprimirMatriz(M);
   printf("V2:\n");
   ImprimirVector(V2,NCOL);
   int resul=Operacion(V1,M,V2);
   printf("Resultado de la operacion: %d\n",resul);
   return;
}

//función main
int main(void){
   Problema();
   getche();
   return(0);
}
```

Solución ejercicio 5

En esta solución utilizamos dos matrices de NxN elementos. La primera contiene los números del panel y la segunda contiene únicamente 0 (casilla oculta) ó 1 (casilla descubierta). Conforme el usuario va introduciendo números debemos ir actualizando las coincidencias en la segunda:

```
//librerías necesarias
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

//Tamaño del tablero
#define N 4

//función que inicializa las matrices
void Rellenar(int M[N][N], int M1[N][N]){
   int i,j;
   for(i=0;i<N;i++){
      for(j=0;j<N;j++){
         //matriz principal con los numeros
         M[i][j]=rand()%(N*N);
         //matriz que indica las casillas acertadas
         M1[i][j]=0;
      }
   }
   return;
}

//función que muestra el tablero al usuario
int Mostrar(int M[N][N], int M1[N][N], int fallos){
   int i,j;

   system("cls");
   printf("Tablero actual\n");

   int V[N];
   for(i=0;i<N;i++) V[i]=0;

   int n=0;
   for(i=0;i<N;i++){
      int s=0;
      for(j=0;j<N;j++){
         if(M1[i][j]==0){
            printf("   *");
         }else{
            printf("%4d",M[i][j]);
            n++;
         }
         s+=M[i][j];
         V[j]+=M[i][j];
      }
      printf("  - %4d\n",s);
   }
   for(i=0;i<N;i++) printf("----");
   printf("\n");
```

```
    for(i=0;i<N;i++){
        printf("%4d",V[i]);
    }
    printf("\nIntentos %d\n",fallos);
    //devolvemos el numero de casillas destapadas
    return(n);
}

//función que actualiza el tablero con el
//número introducido
int Actualizar(int M[N][N], int M1[N][N], int v){
    int i,j;
    int s=0;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            if(M[i][j]==v){
                M1[i][j]=1;
                s=1;
            }
        }
    }
    return(s);
}

//juego del tablero oculto
void Problema(void){
    //utilizamos dos matrices
    int M[N][N],M1[N][N];
    Rellenar(M,M1);
    int fallos=0;
    while(fallos<10 && Mostrar(M,M1,fallos)<N*N){
        printf("\nIntroduzca nuevo valor: ");
        int v;
        scanf("%d",&v);
        if(Actualizar(M,M1,v)==0) fallos++;
    }
    if(fallos==10){
        printf("Lo siento, has perdido\n");
    }else{
        printf("Felicidades, has ganado\n");
    }
    return;
}

//Función principal
int main(void){
    Problema();
    getche();
    return(0);
}
```

Solución ejercicio 6

En esta solución utilizamos una matriz de enteros donde '0' significa que la casilla no ha sido descubierta y además está libre, '1' significa mina y '2' significa que la casilla ha sido descubierta ya (ten en cuenta que no puede haber una casilla descubierta que tenga mina):

```
//librerías necesarias
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//Tamaño del tablero
#define N 8

//Inicialización del tablero
void Inicializar(int T[N][N]){
   int i,j;
   //ponemos todo el tablero a 0
   //0 significa vacío y sin descubrir
   for(i=0;i<N;i++){
      for(j=0;j<N;j++){
         T[i][j]=0;
      }
   }
   //colocamos las minas aleatoriamente
   int nminas=0;
   while(nminas<10){
      i=rand()%N;
      j=rand()%N;
      if(T[i][j]==0){
         //las minas las marcamos como 1
         T[i][j]=1;
         nminas++;
      }
   }
   return;
}

//impresión del tablero
void Imprimir(int T[N][N], int descubierto){
   int i,j;
   for(i=0;i<N;i++){
      for(j=0;j<N;j++){
         switch(T[i][j]){
            case 0:
               printf("? ");
               break;
            case 1:
               if(descubierto==0) printf("? ");
```

```
                else            printf("M ");
                break;
            case 2:
                printf("X ");
                break;
        }
    }
    printf("\n");
  }
  return;
}

//responde si el tablero está o no completo
int Completo(int T[N][N]){
  int i,j;
  int completo=1;
  for(i=0;i<N;i++){
    for(j=0;j<N;j++){
      if(T[i][j]==0) completo=0;
    }
  }
  return(completo);
}

//menu del juego
int Menu(void){
  printf("\nMenu\n");
  printf("1) Tirar\n");
  printf("2) Hacer trampa\n");
  printf("3) Terminar\n");
  int op;
  do{
    printf("Elija opcion (1-3): ");
    scanf("%d",&op);
  }while(op<1 || op>3);
  return(op);
}

//solicita un numero entre 1 y N
int PedirNumero(void){
  int num;
  do{
    scanf("%d",&num);
    if(num<1 || num>N){
      printf("error, introduzca un numero de 1 a %d :",N);
    }
  }while(num<1 || num>N);
  //restamos uno por la numeracion de los vectores
  num=num-1;
  return(num);
```

```
}

//Función para tirar
int Tirar(int T[N][N]){
   int i,j;
   printf("Introduzca fila (1-%d) ",N);
   i=PedirNumero();
   printf("Introduzca columna (1-%d) ",N);
   j=PedirNumero();
   int salida;
   if(T[i][j]==1){
      salida=1; //para que termine el bucle del menu
   }else{
      T[i][j]=2; //en caso contrario tira
      salida=0;
   }
   return(salida);
}

//función principal
void Problema(void){
   int T[N][N],resul;
   Inicializar(T);
   int terminar=0;
   while(terminar==0){
      system("cls");
      printf("JUEGO DEL BUSCAMINAS\n\n");
      Imprimir(T,0);
      int op=Menu();
      switch(op){
         case 1:
            resul=Tirar(T);
            if(resul==1){
               printf("Lo siento, has perdido\n");
               Imprimir(T,1);
               terminar=1;
            }
            break;
         case 2:
            Imprimir(T,1);
            printf("Pulse una tecla\n");
            getche();
            break;
         case 3:
            printf("Adios\n");
            terminar=1;
            break;
      }
      if(Completo(T)==1){
         printf("Felicidades, has ganado\n");
```

```
            terminar=1;
        }
    }
    return;
}

//función main
int main(void){
    Problema();
    getche();
    return(0);
}
```

Solución ejercicio 7

Una posible solución sería:

```
//librerías necesarias
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//tamaño del tablero
#define N 9

//Inicialización del tablero
void Inicializar(int T[N][N]){
    int i,j;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            T[i][j]=0;
        }
    }
    return;
}

//impresión del tablero
void Imprimir(int T[N][N]){
    int i,j;
    for(i=0;i<N;i++){
        if(i!=0 && i%3==0){
            for(j=0;j<21;j++) printf("-");
            printf("\n");
        }
        for(j=0;j<N;j++){
            if(j!=0 && j%3==0) printf("| ");
            if(T[i][j]=='0') printf(" ");
            else printf("%d ",T[i][j]);
        }
        printf("\n");
```

```
    }
    return;
}

//responde si el tablero está o no completo
int Completo(int T[N][N]){
    int i,j;
    int completo=1;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            if(T[i][j]==0) completo=0;
        }
    }
    return(completo);
}

//solicita un número entre 1 y N
int PedirNumero(void){
    int n;
    do{
        scanf("%d",&n);
        if(N<1 || n>N){
            printf("Error: introduzca numero de 1 a %d :",N);
        }
    }while(n<1 || n>N);
    return(n);
}

//comprueba si una tirada es válida
int TiradaValida(int T[N][N],int i1, int j1, int valor){
    int i,j;
    int valido=1;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            if(T[i][j]==valor){
                //coincide en fila o columna
                if(i==i1 || j==j1) valido=0;
                //coincide en cuadrante
                if(i/3==i1/3 && j/3==j1/3) valido=0;
            }
        }
    }
    return(valido);
}

//Función para tirar
void Tirar(int T[N][N]){
    //solicitiamos fila, columna y valor
    printf("Introduzca fila (1-9): ");
    int i=PedirNumero();
```

```
    i--; //se resta uno por la numeracion de vectores
    printf("Introduzca columna (1-9): ");
    int j=PedirNumero();
    j--; //se resta uno por la numeracion de vectores
    printf("Introduzca valor (1-9): ");
    int v=PedirNumero();

    if(TiradaValida(T,i,j,v)==1){
        T[i][j]=v;
    }else{
        printf("Error: tirada no valida\n");
        getche();
    }
    return;
}

//función principal
void Problema(void){
    int T[N][N];
    Inicializar(T);
    while(Completo(T)==0){
        system("cls");
        printf("JUEGO DEL SUDOKU\n\n");
        Imprimir(T);
        printf("\n");
        Tirar(T);
    }
    printf("Felicidades! Has completado el sudoku!\n");
    return;
}

//función main
int main(void){
    Problema();
    getche();
    return(0);
}
```

Solución ejercicio 8

Uno de los inconvenientes mayores de la implementación es generar un panel que no sea imposible de resolver. Para ello, partimos de un panel ordenado y ejecutamos N movimientos aleatorios sobre el mismo, que vendrán dados por el nivel de dificultad. De esta forma, aseguramos que el panel inicial tiene solución.

```
//Librerías necesarias
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

//tamaño del panel
```

```
#define N 4

//función que imprime el panel de juego
void ImprimirPanel(int M[N][N]){
   int i,j;
   printf("\n--- PANEL DE JUEGO ---\n\n");
   for(i=0;i<N;i++){
      for(j=0;j<N;j++){
         if(M[i][j]>=0) printf("%3d ",M[i][j]);
         else           printf("  * ");
      }
      printf("\n\n");
   }
   return;
}

//función que busca el hueco con el *
//(el único número negativo del panel)
void BuscarHueco(int M[N][N], int &posi, int &posj){
   int i,j;
   for(i=0;i<N;i++){
      for(j=0;j<N;j++){
         if(M[i][j]<0){
            posi=i;
            posj=j;
         }
      }
   }
   return;
}

//Funcion que realiza el movimiento
Int Movimiento(int M[N][N], int m){
   int salida=0;
   int i,j;
   BuscarHueco(M,i,j);
   switch(m){
      case 0: //arriba
         if(i>0){
            M[i][j]=M[i-1][j];
            M[i-1][j]=-1;
            salida=1;
         }
         break;
      case 1: //derecha
         if(j<N-1){
            M[i][j]=M[i][j+1];
            M[i][j+1]=-1;
            salida=1;
         }
```

```
            break;
        case 2: //abajo
            if(i<N-1){
                M[i][j]=M[i+1][j];
                M[i+1][j]=-1;
                salida=1;
            }
            break;
        case 3: //izquierda
            if(j>0){
                M[i][j]=M[i][j-1];
                M[i][j-1]=-1;
                salida=1;
            }
            break;
    }
    //devolvemos 1 o 0 en función de si
    //el movimiento ha sido exitoso
    return(salida);
}


void Inicializar(int M[N][N]){
    int nivel, ncambios, i,j;
    printf("Inicializacion del panel\n");

    //nivel de juego
    printf("Nivel (1/2/3): ");
    scanf("%d",&nivel);
    ncambios=N*N*nivel;

    //inicialización en orden
    int act=1;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            M[i][j]=act;
            act++;
        }
    }
    M[N-1][N-1]=-1; //hueco (abajo derecha)

    //generamos los movimientos aleatorios
    while(ncambios>0){
        if(Movimiento(M,rand()%4)==1){
            ncambios--;
        }
    }
    return;
}
```

```
//funcion que comprueba si el panel está solucionado
int Solucionado(int M[N][N]){
   int i,j,soluc=1,act=1;
   for(i=0;i<N;i++){
      for(j=0;j<N;j++){
         if((i<N-1 || j<N-1) && M[i][j]!=act){
            soluc=0;
         }
         act++;
      }
   }
   return(soluc);
}

//funcion que gestiona el juego completo
void Juego(void){
   int M[N][N];
   Inicializar(M);
   char opc;

   do{
      system("cls");
      ImprimirPanel(M);
      printf("\nOpciones:\n");
      printf("  CURSORES: mover hueco\n");
      printf("  ESC     : finalizar\n");
      printf("  R       : inicializar juego\n");
      opc=getche();
      //los cursores devuelven dos caracteres
      //el primero (-32 o 0) es el panel del que
      //provienen (simple o extendido)
      if(opc==-32 || opc==0) opc=getche();
      printf("\n");

      switch(opc){
         case 27: //escape
            break;
         case 72: //arriba
            printf("ARRIBA\n");
            Movimiento(M,0);
            break;
         case 75: //izquierda
            printf("IZQUIERDA\n");
            Movimiento(M,3);
            break;
         case 77: //derecha
            printf("DERECHA\n");
            Movimiento(M,1);
            break;
         case 80: //abajo
```

```
            printf("ABAJO\n");
            Movimiento(M,2);
            break;
         case 'R':
            Inicializar(M);
            break;
      }
   }while(Solucionado(M)!=1 && opc!=27);

   if(opc==27){
      printf("Fin del juego\n");
   }else{
      system("cls");
      ImprimirPanel(M);
      printf("Felicidades, has ganado\n");
   }
   return;
}

//función principal
int  main(void){
   Juego();
   getche();
   return(0);
}
```