```cpp
/*
* Juego de la Serpiente v4
* Pablo_Villa 08/11/2023
*/
#include <iostream>
#include "terminal.h"
#include <cstdlib>
#include <ctime>

using namespace std;

const char TECLA_SIGUIENTE = ' ';
const char TECLA_FIN = 'F';
const char SERPIENTE = '@';
const char CUERPO_SERPIENTE = 'o';
const char ARRIBA = 'W';
const char ABAJO = 'S';
const char IZQUIERDA = 'A';
const char DERECHA = 'D';
const char MANZANA = 'M';
const char SIMBOLO_VERTICAL = '|';
const char SIMBOLO_INTERMEDIO = ' ';
const char SIMBOLO_HORIZONTAl = '-';
const char SIMBOLO_EXTERIOR = '+';
const int PREMIO = 100;
const int LONG_SERPIENTE = 15;
const int BASE = 80;
const int ALTURA = 22;
const int RETARDO = 50;
const int SERPIENTE_X_INICIAL = 10;
const int LIMITE_SUPERIOR = 1;
const int LIMITE_INFERIOR = 20 ;
const int LIMITE_IZQUIERDA = 2;
const int LIMITE_DERECHA = 78;
const int SERPIENTE_Y_INICIAL = 15;
const int MOVIMIENTO_X_DERECHA = 1;
const int MOVIMIENTO_Y_DESCENDENTE = 1;
const int MOVIMIENTO_X_IZQUIERDA = -1;
const int MOVIMIENTO_Y_ASCENDENTE = -1;
const int MAX_MANZANAS = 10;
const string TITULO = "Juego de la serpiente ";
const string VERSION = "4.0";
const string TECLA_CONTINUAR = "ESPACIO";

struct posicion {
    int x = 0;
    int y = 0;
};

struct inc_unitario_posicion {
    int x = 0;
    int y = 0;
};

void iniciar_pantalla_inicial();
void inicializar_juego(char tecla,posicion serpiente[],inc_unitario_posicion& inc_unitario_posicion);
void pantalla_incial();
void dibujar_linea(const char c_exterior, const char c_interior, const int largo);
void dibujar_rectangulo(const int base, const int altura);
void inicializar_serpiente(posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion);
bool juego_terminado(char tecla,posicion serpiente[]);
void obtener_direccion(const char tecla,inc_unitario_posicion& inc_unitario_posicion);
void pintar_serpiente(const posicion serpiente[]);
void borrar_serpiente( const posicion serpiente[]);
bool cuerpo_colisionado(const posicion serpiente[]);
void mover_serpiente(posicion serpiente[], inc_unitario_posicion inc_unitario_posicion);
```

```
67
68  int main() {
69      char tecla = '\0';
70      posicion serpiente[LONG_SERPIENTE];
71      inc_unitario_posicion inc_unitario_posicion;
72      inc_unitario_posicion.x = 0;
73      inc_unitario_posicion.y = 0;
74
75      srand(time(0));
76      setlocale(LC_ALL, "");
77
78      iniciar_pantalla_inicial();
79      while(leer_tecla()!= TECLA_SIGUIENTE){
80              retardar(RETARDO);
81      }
82      inicializar_juego(tecla,serpiente,inc_unitario_posicion);
83
84     while ( ! juego_terminado(tecla, serpiente)) {
85          pintar_serpiente(serpiente);
86
87          retardar(RETARDO);
88
89          borrar_serpiente(serpiente);
90
91          obtener_direccion(tecla, inc_unitario_posicion);
92          mover_serpiente(serpiente, inc_unitario_posicion);
93
94          tecla = leer_tecla();
95      }
96      deshabilitar_modo_crudo_terminal();
97      borrar_terminal();
98  }
99
100 void iniciar_pantalla_inicial(){
101     retardar(RETARDO);
102     hacer_cursor_visible(false);
103     pantalla_incial();
104 }
105
106 void inicializar_juego(char tecla ,posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion){
107     deshabilitar_modo_crudo_terminal();
108     borrar_terminal();
109
110     inicializar_serpiente(serpiente, inc_unitario_posicion);
111
112     dibujar_rectangulo(BASE, ALTURA);
113     retardar(RETARDO);
114
115     habilitar_modo_crudo_terminal();
116     hacer_cursor_visible(false);
117     tecla = leer_tecla();
118 }
119
120 void pantalla_incial(){
121
122     poner_cursor(1,1);
123     cout << " **************************** " << endl;
124     poner_cursor(1,2);
125     cout << " * "<< TITULO << VERSION << " * " << endl;
126     poner_cursor(1,3);
127     cout << " **************************** " << endl;
128     poner_cursor(1,6);
129     cout << "   ______" << endl;
130     poner_cursor(1,7);
131     cout << " _/      \\ " << endl;
132     poner_cursor(1,8);
```

```
133     cout << "  \\___     \\ " << endl;
134     poner_cursor(1,9);
135     cout << "      \\\    \\_____" << endl;
136     poner_cursor(1,10);
137     cout << "       \\\         \\ " << endl;
138     poner_cursor(1,11);
139     cout << "        \\____     \\_____      _|_ " << endl;
140     poner_cursor(1,12);
141     cout << "             \\\         0 \\_/  /   \\ " << endl;
142     poner_cursor(1,13);
143     cout << "              \\__________/ \\  \\___/" << endl;
144     poner_cursor(1,17);
145     cout << "Pulsa la tecla de " <<  TECLA_CONTINUAR << " para continuar" << endl;
146 }
147
148 void inicializar_serpiente(posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion) {
149     serpiente[0].x = SERPIENTE_X_INICIAL;
150     serpiente[0].y = SERPIENTE_Y_INICIAL;
151
152     inc_unitario_posicion.x = MOVIMIENTO_X_DERECHA;
153     inc_unitario_posicion.y = MOVIMIENTO_Y_ASCENDENTE;
154
155     for (int i = 1; i < LONG_SERPIENTE; i++) {
156         serpiente[i].x = serpiente[i - 1].x + 1;
157         serpiente[i].y = serpiente[i - 1].y;
158     }
159 }
160
161 void dibujar_linea(const char c_exterior, const char c_interior, const int largo){
162     cout << c_exterior;
163     for(int i = 0; i < largo - 2; i++){
164         cout << c_interior;
165     }
166     cout << c_exterior << endl;
167 }
168
169 void dibujar_rectangulo(const int base, const int altura){
170     poner_cursor(2,1);
171     cout << "+------------------------- " << TITULO
172         << VERSION << " -------------------------+ " << endl;
173     for (int i = 2; i < altura -2; i++){
174         poner_cursor(2,i);
175         dibujar_linea(SIMBOLO_VERTICAL,SIMBOLO_INTERMEDIO,base);
176     }
177     poner_cursor(2,altura - 2);
178     dibujar_linea(SIMBOLO_EXTERIOR,SIMBOLO_HORIZONTAl,base);
179     poner_cursor(2, altura);
180     cout << ARRIBA << "-> Subir " << ABAJO << "-> Bajar " << IZQUIERDA
181          << "-> Izda " << DERECHA << "-> Dcha " << TECLA_FIN << "-> Fin" << endl;
182 }
183
184 bool juego_terminado(char tecla, posicion serpiente[]){
185     return(toupper(tecla) == TECLA_FIN ||
186            cuerpo_colisionado(serpiente) ||
187            serpiente[0].x == LIMITE_IZQUIERDA ||
188            serpiente[0].x == LIMITE_DERECHA ||
189            serpiente[0].y == LIMITE_SUPERIOR ||
190            serpiente[0].y == LIMITE_INFERIOR);
191 }
192
193 void obtener_direccion(const char tecla, inc_unitario_posicion& inc_unitario_posicion) {
194     switch (toupper(tecla)) {
195         case ARRIBA:
196             inc_unitario_posicion.x = 0;
197             inc_unitario_posicion.y = MOVIMIENTO_Y_ASCENDENTE;
198             break;
```

```
199
200         case ABAJO:
201             inc_unitario_posicion.x = 0;
202             inc_unitario_posicion.y = MOVIMIENTO_Y_DESCENDENTE;
203             break;
204
205         case IZQUIERDA:
206             inc_unitario_posicion.x = MOVIMIENTO_X_IZQUIERDA;
207             inc_unitario_posicion.y = 0;
208             break;
209
210         case DERECHA:
211             inc_unitario_posicion.x = MOVIMIENTO_X_DERECHA;
212             inc_unitario_posicion.y = 0;
213             break;
214   }
215 }
216
217 void mover_serpiente(posicion serpiente[], inc_unitario_posicion inc_unitario_posicion) {
218     posicion cabeza_anterior;
219     cabeza_anterior = serpiente[0];
220
221     serpiente[0].x = serpiente[0].x + inc_unitario_posicion.x;
222     serpiente[0].y = serpiente[0].y + inc_unitario_posicion.y;
223
224     for (int i = LONG_SERPIENTE - 1; i > 0; --i) {
225         serpiente[i] = serpiente[i - 1];
226     }
227     serpiente[1] = cabeza_anterior;
228 }
229
230
231 void pintar_serpiente(const posicion serpiente[]) {
232     poner_cursor(serpiente[0].x, serpiente[0].y);
233     cout << SERPIENTE;
234
235     for (int i = 1; i < LONG_SERPIENTE - 1; i++) {
236         poner_cursor(serpiente[i].x, serpiente[i].y);
237         cout << CUERPO_SERPIENTE;
238     }
239 }
240
241 void borrar_serpiente( const posicion serpiente[]) {
242     for (int i = 0; i < LONG_SERPIENTE - 1; i++) {
243         poner_cursor(serpiente[i].x, serpiente[i].y);
244         cout << " ";
245     }
246 }
247
248 bool cuerpo_colisionado(const posicion serpiente[]) {
249     for (int i = 1; i < LONG_SERPIENTE - 1; ++i) {
250         if (serpiente[0].x == serpiente[i].x &&
251             serpiente[0].y == serpiente[i].y) {
252             return true;
253         }
254     }
255     return false;
256 }
```

```cpp
/*
* Juego de la Serpiente v1
* Pablo_Villa 08/11/2023
*/
#include <iostream>
#include "terminal.h"
#include <cstdlib>
#include <ctime>

using namespace std;

const char TECLA_SIGUIENTE = ' ';
const char TECLA_FIN = 'F';
const char SERPIENTE = '@';
const char CUERPO_SERPIENTE = 'o';
const char ARRIBA = 'W';
const char ABAJO = 'S';
const char IZQUIERDA = 'A';
const char DERECHA = 'D';
const char MANZANA = 'M';
const char SIMBOLO_VERTICAL = '|';
const char SIMBOLO_INTERMEDIO = ' ';
const char SIMBOLO_HORIZONTAl = '-';
const char SIMBOLO_EXTERIOR = '+';
const int PREMIO = 100;
const int LONG_SERPIENTE = 15;
const int BASE = 80;
const int ALTURA = 22;
const int RETARDO = 50;
const int SERPIENTE_X_INICIAL = 10;
const int LIMITE_SUPERIOR = 1;
const int LIMITE_INFERIOR = 20 ;
const int LIMITE_IZQUIERDA = 2;
const int LIMITE_DERECHA = 78;
const int SERPIENTE_Y_INICIAL = 15;
const int MOVIMIENTO_X_DERECHA = 1;
const int MOVIMIENTO_Y_DESCENDENTE = 1;
const int MOVIMIENTO_X_IZQUIERDA = -1;
const int MOVIMIENTO_Y_ASCENDENTE = -1;
const int MARGEN_INI_MANZANA = 5;
const int MARGEN_MARCADOR = 5;
const int MAX_MANZANAS = 10;
const string TITULO = "Juego de la serpiente ";
const string VERSION = "5.0";
const string TECLA_CONTINUAR = "ESPACIO";

struct posicion {
    int x = 0;
    int y = 0;
};

struct inc_unitario_posicion {
    int x = 0;
    int y = 0;
};

void iniciar_pantalla_inicial();
void inicializar_juego(char tecla,posicion serpiente[],posicion& manzana,inc_unitario_posicion&
inc_unitario_posicion);
void pantalla_incial();
void dibujar_linea(const char c_exterior, const char c_interior, const int largo);
void dibujar_rectangulo(const int base, const int altura);
void inicializar_serpiente(posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion);
bool juego_terminado(char tecla,posicion serpiente[]);
void obtener_direccion_serpiente(const char tecla,inc_unitario_posicion& inc_unitario_posicion);
void pintar_serpiente(const posicion serpiente[]);
```

```
void borrar_serpiente( const posicion serpiente[]);
bool cuerpo_colisionado(const posicion serpiente[]);
void inicializar_manzana(posicion& manzana);
void pintar_manzana(const posicion& manzana);
void borrar_manzana(const posicion& manzana);
bool manzana_tocada(const posicion& manzana, const posicion serpiente[]);
void mover_serpiente(posicion serpiente[], inc_unitario_posicion inc_unitario_posicion);
void actualizar_marcador(int& puntos);

int main() {
    int puntos = 0;
    bool hay_manzana = false;
    char tecla = '\0';
    posicion serpiente[LONG_SERPIENTE];
    posicion manzana;
    manzana.x = 0;
    manzana.y = 0;
    inc_unitario_posicion inc_unitario_posicion;
    inc_unitario_posicion.x = 0;
    inc_unitario_posicion.y = 0;

    srand(time(0));
    setlocale(LC_ALL, "");

    iniciar_pantalla_inicial();
    while(leer_tecla()!= TECLA_SIGUIENTE){
            retardar(RETARDO);
    }
    inicializar_juego(tecla,serpiente,manzana,inc_unitario_posicion);
    while ( ! juego_terminado(tecla, serpiente)) {

        if( ! hay_manzana){
            pintar_manzana(manzana);
            hay_manzana = true;
        }
        if(manzana_tocada(manzana, serpiente)){
            borrar_manzana(manzana);
            hay_manzana = false;

            inicializar_manzana(manzana);
            actualizar_marcador(puntos);
        }

        pintar_serpiente(serpiente);

        retardar(RETARDO);

        borrar_serpiente(serpiente);

        obtener_direccion_serpiente(tecla, inc_unitario_posicion);
        mover_serpiente(serpiente, inc_unitario_posicion);

        tecla = leer_tecla();
    }
    deshabilitar_modo_crudo_terminal();
    borrar_terminal();
}

void iniciar_pantalla_inicial(){
    retardar(RETARDO);
    hacer_cursor_visible(false);
    pantalla_incial();
}

void inicializar_juego(char tecla, posicion serpiente[], posicion& manzana,inc_unitario_posicion&
inc_unitario_posicion){
```

```
    deshabilitar_modo_crudo_terminal();
    borrar_terminal();

    inicializar_serpiente(serpiente, inc_unitario_posicion);
    inicializar_manzana(manzana);

    dibujar_rectangulo(BASE, ALTURA);
    retardar(RETARDO);

    habilitar_modo_crudo_terminal();
    hacer_cursor_visible(false);
    tecla = leer_tecla();
}

void pantalla_incial(){

    poner_cursor(1,1);
    cout << " **************************** " << endl;
    poner_cursor(1,2);
    cout << " * "<< TITULO << VERSION << " * " << endl;
    poner_cursor(1,3);
    cout << " **************************** " << endl;
    poner_cursor(1,6);
    cout << "   ______" << endl;
    poner_cursor(1,7);
    cout << " _/      \\ " << endl;
    poner_cursor(1,8);
    cout << "  \\___    \\ " << endl;
    poner_cursor(1,9);
    cout << "      \\    \\_____" << endl;
    poner_cursor(1,10);
    cout << "       \\         \\ " << endl;
    poner_cursor(1,11);
    cout << "        \\____     \\______      _|_ " << endl;
    poner_cursor(1,12);
    cout << "             \\         0 \\_/  /   \\ " << endl;
    poner_cursor(1,13);
    cout << "              \\__________/ \\  \\___/" << endl;
    poner_cursor(1,17);
    cout << "Pulsa la tecla de " <<  TECLA_CONTINUAR << " para continuar" << endl;
}

void inicializar_serpiente(posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion) {
    serpiente[0].x = SERPIENTE_X_INICIAL;
    serpiente[0].y = SERPIENTE_Y_INICIAL;

    inc_unitario_posicion.x = MOVIMIENTO_X_DERECHA;
    inc_unitario_posicion.y = MOVIMIENTO_Y_ASCENDENTE;

    for (int i = 1; i < LONG_SERPIENTE; i++) {
        serpiente[i].x = serpiente[i - 1].x + 1;
        serpiente[i].y = serpiente[i - 1].y;
    }
}

void dibujar_linea(const char c_exterior, const char c_interior, const int largo){
    cout << c_exterior;
    for(int i = 0; i < largo - 2; i++){
        cout << c_interior;
    }
    cout << c_exterior << endl;
}

void dibujar_rectangulo(const int base, const int altura){
    poner_cursor(2,1);
    cout << "+------------------------ " << TITULO
```

```
        << VERSION << " -------------------------+ " << endl;
    for (int i = 2; i < altura -2; i++){
        poner_cursor(2,i);
        dibujar_linea(SIMBOLO_VERTICAL,SIMBOLO_INTERMEDIO,base);
    }
    poner_cursor(2,altura - 2);
    dibujar_linea(SIMBOLO_EXTERIOR,SIMBOLO_HORIZONTAl,base);
    poner_cursor(2, altura);
    cout << ARRIBA << "-> Subir " << ABAJO << "-> Bajar " << IZQUIERDA
         << "-> Izda " << DERECHA << "-> Dcha " << TECLA_FIN << "-> Fin" << endl;
}

bool juego_terminado(char tecla, posicion serpiente[]){
    return(toupper(tecla) == TECLA_FIN ||
           cuerpo_colisionado(serpiente) ||
           serpiente[0].x == LIMITE_IZQUIERDA ||
           serpiente[0].x == LIMITE_DERECHA ||
           serpiente[0].y == LIMITE_SUPERIOR ||
           serpiente[0].y == LIMITE_INFERIOR);
}

void obtener_direccion_serpiente(const char tecla, inc_unitario_posicion& inc_unitario_posicion) {
    switch (toupper(tecla)) {
        case ARRIBA:
            inc_unitario_posicion.x = 0;
            inc_unitario_posicion.y = MOVIMIENTO_Y_ASCENDENTE;
            break;

        case ABAJO:
            inc_unitario_posicion.x = 0;
            inc_unitario_posicion.y = MOVIMIENTO_Y_DESCENDENTE;
            break;

        case IZQUIERDA:
            inc_unitario_posicion.x = MOVIMIENTO_X_IZQUIERDA;
            inc_unitario_posicion.y = 0;
            break;

        case DERECHA:
            inc_unitario_posicion.x = MOVIMIENTO_X_DERECHA;
            inc_unitario_posicion.y = 0;
            break;
  }
}

void mover_serpiente(posicion serpiente[], inc_unitario_posicion inc_unitario_posicion) {
    posicion cabeza_anterior;
    cabeza_anterior = serpiente[0];

    serpiente[0].x = serpiente[0].x + inc_unitario_posicion.x;
    serpiente[0].y = serpiente[0].y + inc_unitario_posicion.y;

    for (int i = LONG_SERPIENTE - 1; i > 0; --i) {
        serpiente[i] = serpiente[i - 1];
    }
    serpiente[1] = cabeza_anterior;
}


void pintar_serpiente(const posicion serpiente[]) {
    poner_cursor(serpiente[0].x, serpiente[0].y);
    cout << SERPIENTE;

    for (int i = 1; i < LONG_SERPIENTE - 1; i++) {
        poner_cursor(serpiente[i].x, serpiente[i].y);
        cout << CUERPO_SERPIENTE;
```

```
    }
}

void borrar_serpiente( const posicion serpiente[]) {
    for (int i = 0; i < LONG_SERPIENTE - 1; i++) {
        poner_cursor(serpiente[i].x, serpiente[i].y);
        cout << " ";
    }
}

bool cuerpo_colisionado(const posicion serpiente[]) {
    for (int i = 1; i < LONG_SERPIENTE - 1; ++i) {
        if (serpiente[0].x == serpiente[i].x &&
            serpiente[0].y == serpiente[i].y) {
            return true;
        }
    }
    return false;
}

void inicializar_manzana(posicion& manzana) {
    manzana.x = LIMITE_IZQUIERDA + MARGEN_INI_MANZANA +
    rand() % (LIMITE_DERECHA - LIMITE_IZQUIERDA - MARGEN_INI_MANZANA );

    manzana.y = LIMITE_SUPERIOR + MARGEN_INI_MANZANA +
    rand() % (LIMITE_INFERIOR - LIMITE_SUPERIOR - MARGEN_INI_MANZANA);
}

void pintar_manzana(const posicion& manzana){
    poner_cursor(manzana.x, manzana.y);
    cout << MANZANA;
}

void borrar_manzana(const posicion& manzana){
    poner_cursor(manzana.x, manzana.y);
    cout << " ";
}

bool manzana_tocada(const posicion& manzana, const posicion serpiente[]){
    return(manzana.x == serpiente[0].x &&
           manzana.y == serpiente[0].y);
}

void actualizar_marcador(int& puntos){
    puntos = puntos + PREMIO;
    poner_cursor(LIMITE_IZQUIERDA,LIMITE_INFERIOR + MARGEN_MARCADOR);
    cout << "PUNTOS: " << puntos;
}
```

Ejercicio 3

Tanto en el primer como en el segundo caso se produce una operación con un resultado infinito. En el primer caso obtenemos inf, por lo que nos muestra inf de infinito positivo. En el segundo caso obtenemos –inf, por lo que nos muestra –inf de infinito negativo.

En el tercer caso obtenemos NaN, que significa “Not an Number”. Obtenemos este resultado debido a un error de representación de reales. Esto es debido a que el segundo factor de la división no puede ser un 0 para poder realizar una operación, porque no existe. En nuestro caso si tenemos ese 0, por lo que obtenemos ese resultado NaN.

En el cuarto y último caso, no obtenemos nada. Al estar dividiendo 0 entre 0, operación que no existe, el resultado tampoco existe, por eso no obtenemos nada al mostrarlo por pantalla. La diferencia con el caso anterior reside en que en el cuarto caso la operación no está definida, por eso no es posible mostrar el resultado por pantalla

```cpp
/*
 * Pablo_Villa 874773
 * 01/12/23
 */

#include <iostream>
using namespace std;

const int MAX_PRODUCTOS = 100;
const int ALTA = 1;
const int BAJA = 2;
const int LISTADO = 3;
const int ENTRADA = 4;
const int SALIDA = 5;
const int BUSCAR = 6;
const int FIN = 7;

struct producto_almacen {
    int codigo = 0;
    string descripcion = "";
    int existencias = 0;
};

void anadir_producto(producto_almacen productos[], int &numero_productos);
void leer_producto(producto_almacen &producto);
void listar_productos(producto_almacen productos[], int numero_productos);
void anadir(producto_almacen productos[], int &numero_productos);
void eliminar(producto_almacen productos[], int &numero_productos);
void borrar_producto(producto_almacen productos[], int &numero_productos);
void buscar(producto_almacen productos[], int &numero_productos);
int busqueda(producto_almacen productos[], int codigo, int numero_productos);

int main(){
    producto_almacen productos[MAX_PRODUCTOS];
    int opcion = 0;
    int numero_productos = 0;

    while(opcion != FIN){
        cout << ALTA << "->Alta " << BAJA << "->Baja " << LISTADO << "->Listado " << ENTRADA
             << "->Entrada " << SALIDA << "->Salida " << BUSCAR << "-> Buscar " << FIN << "->Acabar: ";

        cin >> opcion;

        switch(opcion){
            case ALTA:
                anadir_producto(productos, numero_productos);
                break;

            case BAJA:
                borrar_producto(productos, numero_productos);
                break;

            case LISTADO:
                listar_productos(productos, numero_productos);
                break;

            case ENTRADA:
                anadir(productos, numero_productos);
                break;

            case SALIDA:
                eliminar(productos, numero_productos);
                break;

            case BUSCAR:
                buscar(productos, numero_productos);
```

```
                break;

            case FIN:
                cout << "Fin" << endl;
                break;
        }
    }
}

void leer_producto(producto_almacen &producto){
    cout << "Código : ";
    cin >> producto.codigo;
    cout << "Descripción : ";
    cin.ignore();
    getline(cin, producto.descripcion);
    cout << "Existencias : ";
    cin >> producto.existencias;
}

void anadir_producto(producto_almacen productos[], int &numero_productos){
    if (numero_productos > MAX_PRODUCTOS){
        cout << "Número máximo de productos" << endl;
    } else {
        leer_producto(productos[numero_productos]);
        numero_productos++;
    }
}

void borrar_producto(producto_almacen productos[], int &numero_productos){
    int codigo = 0;
    int posicion = 0;
    cout << "Introduce un código de un producto que desees eliminar ";
    cin >> codigo;


    posicion = busqueda(productos, codigo, numero_productos);
    if (posicion != -1) {
        productos[posicion] = productos[numero_productos - 1];
        numero_productos--;
        cout << "Producto eliminado" << endl;
    }
    else {
        cout << "Producto no encontrado" << endl;
    }
}

void mostrar_producto(const producto_almacen &producto){
    cout << "Codigo: ";
    cout << producto.codigo << endl;
    cout << "Descripcion: ";
    cout << producto.descripcion << endl;
    cout << "Existencias: ";
    cout << producto.existencias << endl;
}

void listar_productos(producto_almacen productos[], int numero_productos){
    for (int i = 0; i < numero_productos; i++){
        mostrar_producto(productos[i]);
    }
}

void anadir(producto_almacen productos[], int &numero_productos){
    int codigo = 0;
    int existencias = 0;
    bool encontrado = false;
    int posicion = 0;
```

```
133
134     cout << "codigo: ";
135     cin >> codigo;
136     cout << "Añadir existencias: ";
137     cin >> existencias;
138
139     posicion = busqueda(productos, codigo, numero_productos);
140     if (posicion == -1) {
141         cout << "El código introducido no corresponde con ningún producto." << endl;
142     } else {
143         productos[posicion].existencias += existencias;
144         cout << "Se han almacenado (" << existencias << ") de "
145              << productos[posicion].descripcion << " con código " << productos[posicion].codigo << endl;
146     }
147 }
148
149 void eliminar(producto_almacen productos[], int &numero_productos){
150     int codigo = 0;
151     int existencias = 0;
152     bool encontrado = false;
153     int posicion = 0;
154     cout << "Codigo: ";
155     cin >> codigo;
156     cout << "Eliminar existencias: ";
157     cin >> existencias;
158
159     posicion = busqueda(productos, codigo, numero_productos);
160     if (posicion == -1) {
161         cout << "El código introducido no corresponde con ningún producto." << endl;
162     } else {
163         productos[posicion].existencias -= existencias;
164         cout << "Se han retirado (" << existencias << ") de "
165              << productos[posicion].descripcion << " con código " << productos[posicion].codigo << endl;
166     }
167 }
168
169 void buscar(producto_almacen productos[], int &numero_productos){
170     int codigo = 0;
171     int posicion = 0;
172     cout << "Introduce un codigo que desees buscar ";
173     cin >> codigo;
174
175     posicion = busqueda(productos, codigo, numero_productos);
176     if(posicion == -1){
177         cout << "El código introducido no corresponde a ningún producto" << endl;
178     } else {
179         mostrar_producto(productos[posicion]);
180     }
181 }
182
183 int busqueda(producto_almacen productos[], int codigo, int numero_productos){
184     for (int i = 0; i < numero_productos; i++){
185         if(codigo == productos[i].codigo){
186             return i;
187         }
188     }
189     return -1;
190 }
```